PRIME

Initial Documentation Release

IDR 3046
PRIME COMPUTER
COBOL
REFERENCE GUIDE FOR DBMS

P/N IDR 3062

First Printing July 1977

CONTENTS

ILLUSTRATIONS

TABLES

FOREWORD

The COBOL Reference Manual for DBMS contains specifications of the Data Description Language declaring a COBOL sub-schema (Section 2), and the COBOL Data Manipulation Language (Section 3).

DOCUMENTATION EXCELLENCE

Prime is striving to maintain the highest documentation standards. To achieve this goal, the Database documentation will be published in three documentation releases as described in Section 1. This is the Initial Documentation Release. Prime asks that each serious Database user correspond his comments about this manual concerning technical accuracy and additional information needed to implement the task of Database Administrator.

Robert E. Dawes, Technical Writer
Technical Publications Department
Prime Computer Inc.
145 Pennsylvania Avenue,
Framingham, Ma. 01701

SECTION 1

INTRODUCTION

ABOUT THIS MANUAL

This manual is oriented toward knowledgeable database application programmers. The reader is assumed to be acquainted with basic concepts of virtual memory operating systems and familiar with the benefits of database management principles.

This manual contains reference information for the COBOL Data Description Language (DDL) and Data Manipulation Language (DML). Section 2 contains the language specifications for COBOL DDL and Section 3 contains language specifications for COBOL DML.

DATABASE DOCUMENTATION

Database documentation (Figure 1-1) is provided for both the Database Administrator and the application programmer. The Database Administrator uses two manuals: 1) The Prime Database Administrator User's Guide and 2) Prime DBMS Schema Data Description Language (DDL) Reference Manuals.

The application programmer uses two manuals per application language: 1) the Prime FORTRAN Reference Manual for DBMS and the Companion Prime FORTRAN User's Guide, 2) The COBOL Reference Manual for DBMS and the Companion Prime COBOL User's Manual.

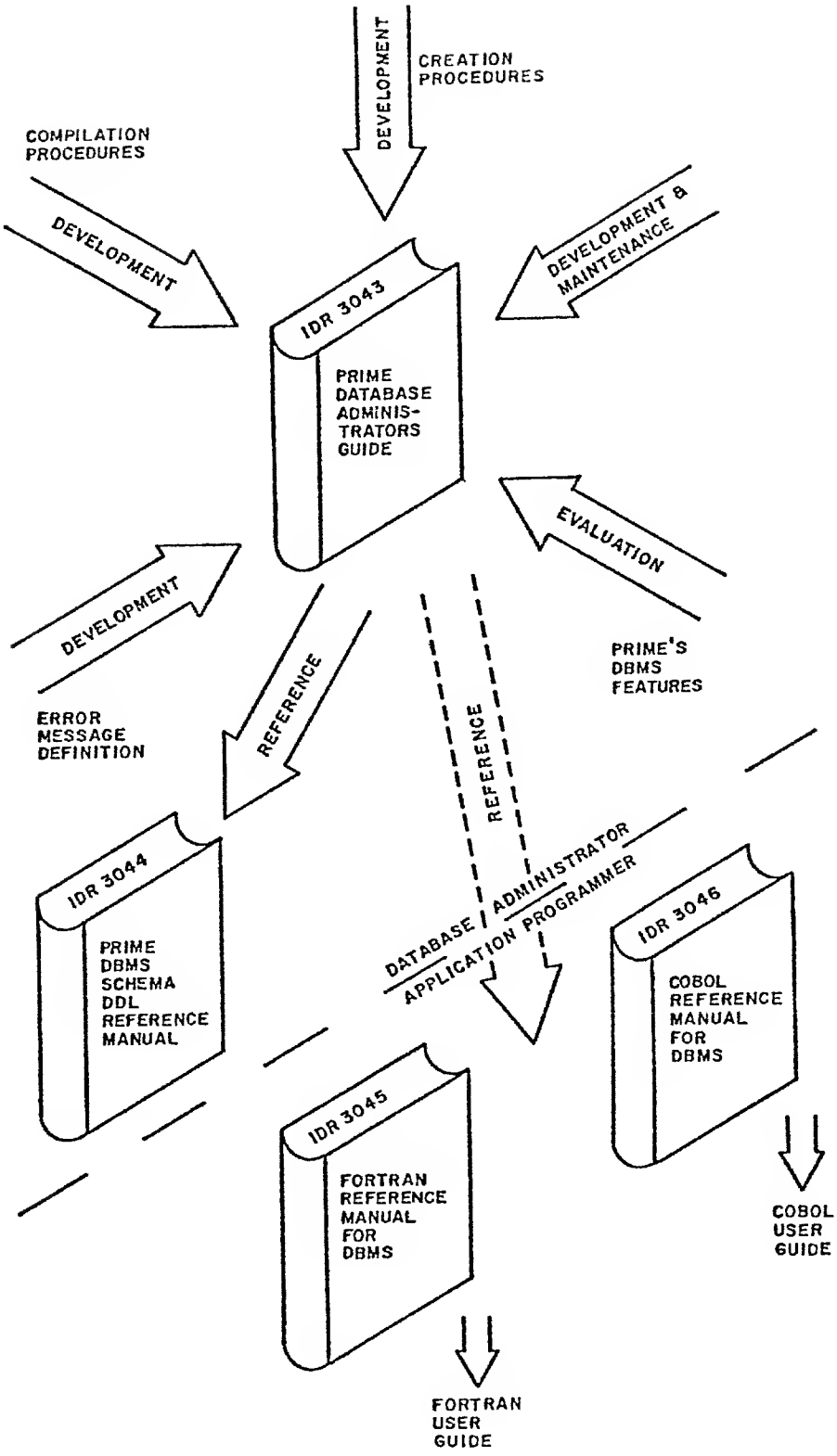


Figure 1-1. PRIME DBMS DOCUMENTATION

Documentation Releases

Prime provides three documentation releases (see Figure 1-2) for every new product: The Initial Documentation Release (IDR), the Preliminary Documentation Release (PDR), and the Final Documentation Release (FDR).

The Initial Documentation Release (IDR) provides whatever information is available without regard to whether the information is grammatically correct, or properly formatted. Thus, the intent is to provide usable information when needed.

The Preliminary Documentation Release (PDR) is the second draft by the writer. It provides more complete and accurate information about the product, but does not represent the final product information. Customers having the IDR should request the PDR from his sales representative.

The Final Documentation Release (FDR) is the complete product description up to the stated software revision number. This release is edited, formatted and presented in Prime's highest professional standards. Users will be notified when this release is available and should contact the local sales representative for a copy.

OTHER RELATED PRIME DOCUMENTS

PRIMOS FILE SYSTEM USER'S GUIDE	MAN2604
PRIMOS INTERACTIVE USER'S GUIDE	MAN2602
PRIMOS COMPUTER ROOM USER'S GUIDE	MAN2603
PROGRAM DEVELOPMENT SOFTWARE USER'S GUIDE	MAN1879
COBOL IV USER'S GUIDE	MAN3057
COBOL USER'S GUIDE	MAN2797

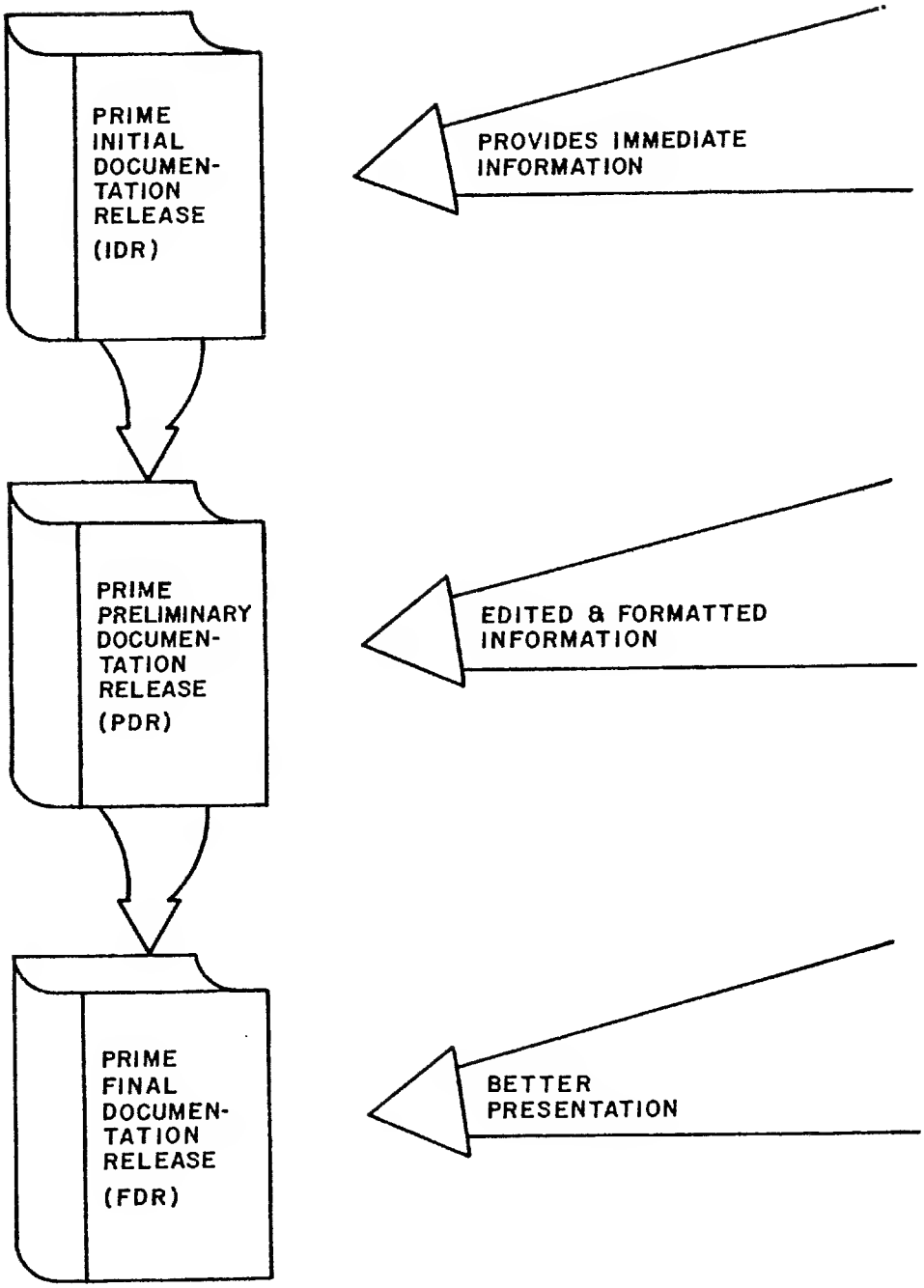


Figure 1-2. DOCUMENTATION RELEASES

THE COBOL/DBMS INTERFACE

The COBOL interface to the DBMS includes two major processors and their respective languages: the COBOL Subschema Data Definition Language (DDL) Compiler and the COBOL Data Manipulation Language (DML) Preprocessor.

The application programmer's "view" of a schema is written in the COBOL Subschema DDL. The Subschema Compiler translates the DDL into an internal, tabular form called the subschema table which is used by the DML Preprocessor.

Commands for locating a string, as well as retrieving, deleting, and modifying the contents of a database, are written in the COBOL DML. These commands are interspersed with COBOL statements in the application program source file and translated into COBOL declarations and statements by the COBOL DML Preprocessor. The output of the Preprocessor is the source input for the COBOL Compiler.

DATABASE DEVELOPMENT

The database development sequence of events is illustrated in Figure 1-3. Before performing any compilation, the user should examine this flow to assure that the database is constructed in the proper sequence.

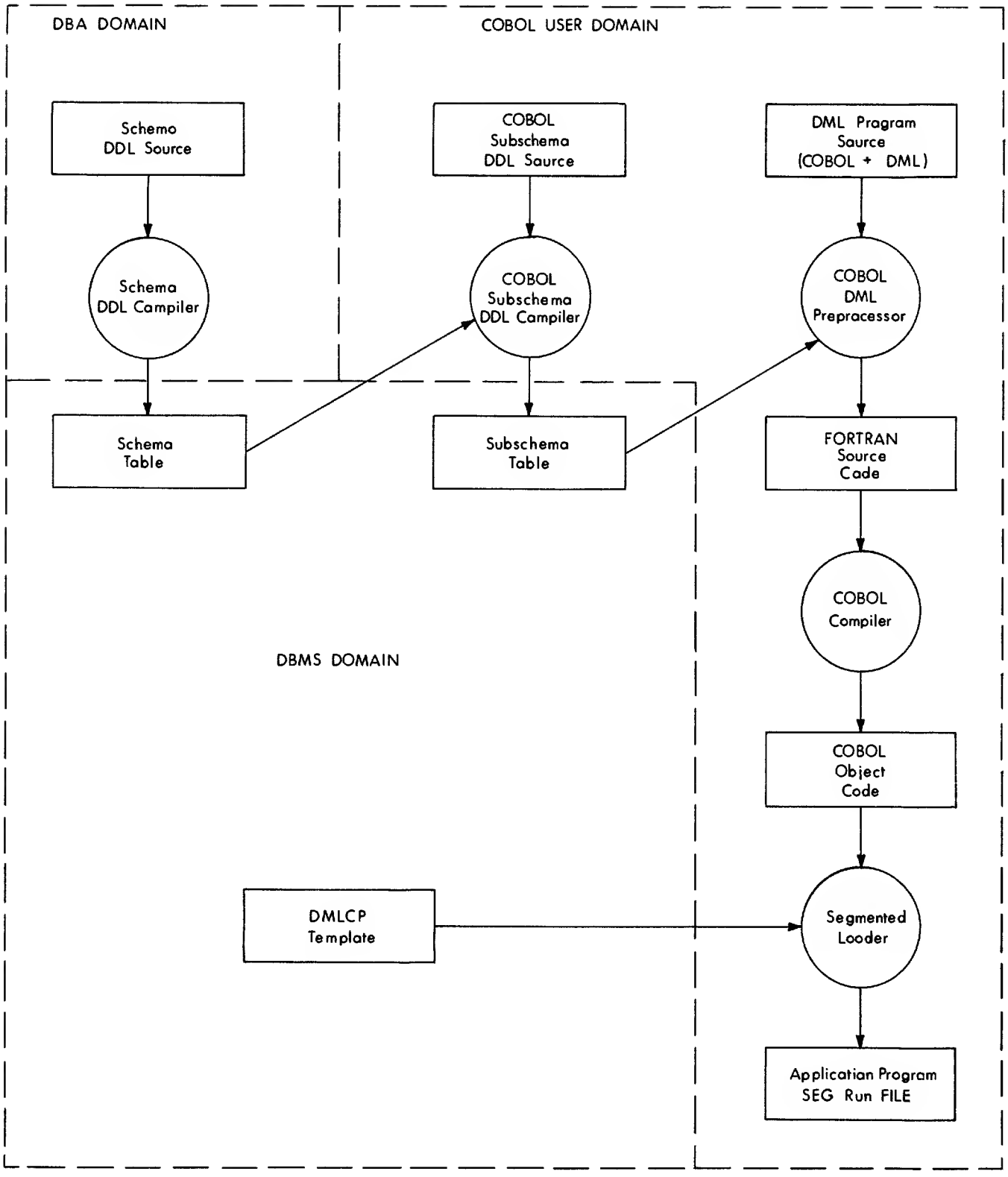


Figure 1-3 Database Development Sequence of Events

SECTION 2

COBOL DDL

INTRODUCTION

This section contains the specifications of the Data Description Language (DDL) for declaring a COBOL subschema.

The specifications provide for two divisions which serve to:

- Identify the subschema (subschema IDENTIFICATION division).
- Define (and optionally rename) the areas, records, and sets included in the subschema (subschema DATA division).

The subschema IDENTIFICATION division must precede the subschema DATA division and consists of a single entry.

CODING INSTRUCTIONS

Character Set

The COBOL subschema data source language character set consists of the following characters:

Letters A through Z
Blank or space
Digits Ø through 9
Special characters:
+ Plus sign
- Minus sign
* Asterisk
= Equal sign
> Relational sign (greater than)
< Relational sign (less than)
$ Dollar sign
, Comma
; Semicolon
. Period or decimal point
" Quotation mark
(Left parenthesis
) Right parenthesis
' Apostrophe (alternate of quotation mark)
/ Virgule

Of the previous set, the following characters are used for words:

Ø through 9
A through Z
- (hyphen)

The following characters are used for punctuation:

(Left parenthesis
) Right parenthesis
, Comma
. Period
; Semicolon

The following relational characters are used in simple conditions:

>
<
=

Punctuation

The following general rules of punctuation apply in writing source programs:

1. A period, semicolon, or comma, when used, should not be preceded by a space, but must be followed by a space.

2. A left parenthesis should not be followed immediately by a space; a right parenthesis should not be preceded immediately by a space.

3. At least one space must appear between two successive words and/or literals. Two or more successive spaces are treated as single space, except in non-numeric literals.

4. Relation characters should always be preceded by a space and followed by another space.

5. When the period, comma, plus, or minus characters are used in the PICTURE clause, they are governed solely by rules for report items.

6. A comma may be used as a separator between successive operands of a statement, or between two subscripts.

7. A semicolon or comma may be used to separate a series of statements or clauses.

Word Formation

A word is composed of a combination of not more than 30 characters, chosen from the following set of 37 characters:

0 through 9 (digits)
A through Z (letters)
- (hyphen)

A word must begin with a letter; it may not end with a hyphen. A word is ended by a space, or by proper punctuation. A word may contain more than one embedded hyphen; consecutive embedded hyphens are also permitted. All words are either reserved words, (refer to Appendix) which have preassigned meanings, or programmer-supplied names. If a programmer-supplied name is not unique, there must be a unique method of reference to it by use of name qualifiers; e.g., TAX-RATE IN STATE-TABLE. Primarily, a non-reserved word identifies a data item or field, and is called a data-name. Other cases of non-reserved words are file-names, condition-names, mnemonic-names, and procedure-names.

CODING RULES

Since Prime COBOL is a subset of American National Standards Institute (ANSI), COBOL subschemas are written on standard COBOL coding sheets . The following rules are applicable.

1. Each line of code should have a six-digit sequence number in columns 1-6, such that the punched cards are in ascending order. Blanks are also permitted in columns 1-6.

2. Reserved words for division, section, and paragraph headers must begin in Area A (columns 8-11). Level numbers may appear in Margin A.

3. All other program elements must be confined to columns 12-72, governed by the other rules of statement punctuation.

4. Columns 73-80 are ignored by the compiler. Frequently, these columns are used to contain the deck identification.

5. Explanatory comments may be inserted on any line within a source program by placing an asterisk (*) in column 7 of the line. Any combination of characters may be included in Areas A and B of that line. The asterisk and the characters will be produced on the source listing but serve no other purpose. If a slash (/) appears in column 7, the next card will be printed at the top of a new page when the compiler lists the program. A hyphen (-) is used to continue a non-numeric literal from one line to another. Refer to Non-Numeric Literals for coding rules.

FORMAT NOTATION

Throughout this publication, basic formats are prescribed for various clauses or statements. These generalized descriptions guide the programmer in writing his own statements. They are presented in a uniform system of notation:

1. All words printed entirely in capital letters are reserved words. (See Appendix A.) These are words that have preassigned meanings. In all formats, words in capital letters represent an actual occurrence of those words.

2. All underlined reserved words are required unless the portion of the format containing them is itself optional. These are key words. If any key word is missing or is incorrectly spelled, it is considered an error in the program. Reserved words not underlined may be included or omitted at the option of the programmer. These words are optional words; they are used solely for improving readability of the program.

3. The characters < > = when appearing in formats, although not underlined, are required when such formats are used.

4. All punctuation and other special characters represent the actual occurrence of those characters. Punctuation is essential where it is shown. Additional punctuation can be inserted, according to the rules for punctuation specified in this publication. In general, terminal periods are shown in formats in the manual because they are required; semicolons and commas are not shown generally because they are optional.

5. Words printed in lower-case letters in formats represent generic parts (e.g., data-names) of which a valid representation must appear.

6. Parts of a statement or data description entry which are enclosed in brackets ([ ]) are optional. Parts between matching braces ({}) represent a choice of mutually exclusive options of which one must be chosen.

7. Certain entries in the formats consist of a capitalized word(s) followed by the word "Clause" or "Statement." These designate clauses or statements that are described in other formats in appropriate sections of the text.

8. In order to facilitate reference to them in the text, some lower-case words are followed by a hyphen and a digit or letter. This modification does not change the syntactical definition of the word.

9. The ellipsis (...) indicates that the immediately preceding unit may occur once, or any number of times in succession. A unit means either a single lower-case word, or a group of lower-case

words and one or more reserved words enclosed in brackets or braces. If a term is enclosed in brackets or braces, the entire unit of which it is part must be repeated when repetition is specified.

10. Comments, restrictions, and clarifications on the use and meaning of every format are contained in the appropriate portions of this manual.

USING THE COMPILER

The COBOL Subschema DDL Compiler (CSUBS) translates the COBOL Subschema Data Definition Language (DDL) into an internal tabular form called the Subschema table. CSUBS also produces an output listing which includes the DDL text, error messages, and a map of the User Work Area. The listing is found in the current UFD and is called "L<-name", where "name" is the DDL input file name.

Invoking the Compiler

The COBOL Subschema DDL Compiler is invoked withthe command:

```
CSUBS source-file [-VOL volume-name]
```

If volume-name is not specified for the output subschema table, the volume of the schema table is assumed.

Compiler Errors

There are three classes of errors that may be encountered by the compiler:

1. Fatal - cannot parse further.
2. Nonfatal - can continue parsing for further errors after end of this clause, but final subschema table will not be produced.
3. Warning - Syntax is ambiguous or incomplete and the compiler has made an arbitrary decision so that a final Subschema table may be produced.

Error messages are displayed both on theuser terminal with the line in which the error occured, as well as in the output listing after the erroneous line.

User Work Area Map

A map of the User Work Area WORKING STORAGE entries are included in the output listing of the COBOL Subschema DDL Compiler. The map includes

the name, size, COBOL data type, and starting address in User Work Area for all items and data-base-dasta-names specified in the Subschema as well as the various DML run-time error and exception registers.

```
****************************
* SUBSCHEMA IDENTIFICATION *
****************************
```

Function

To define and name a subschema within a schema.

Complete Entry

SUBSCHEMA NAME IS Sub-schema-name OF SCHEMA NAME schema-name

[PRIVACY KEY FOR COPY {literal-1}].

****
* IDENTIFICATION-SUBSCHEMA NAME *
****

Function

To name a subschema and its associated schema.

General Format

SUBSCHEMA NAME IS sub-schema-name OF SCHEMA NAME schema-name.

Syntax Rules

1. Subschema-name must be unique among the subschema-names associated with the specified schema.

General Rules

None

```
******************************
* IDENTIFICATION-PRIVACY KEY *
******************************
```

Function

To specify the key for accessing a schema which includes a privacy lock on its use for developing a subschema (that is, it includes a PRIVACY LOCK FOR COPY clause).

General Format

PRIVACY KEY FOR COPY IS {literal-1}.

Syntax Rules

None

General Rules

None

****
* SUBSCHEMA DATA DIVISION *
****

Function

To name and give certain characteristics of the areas, records, and sets of the schema that are contained in the subschema.

Structure of the Subschema DATA DIVISION

The Subschema DATA consists of four sections. The names of these four sections in their required order of appearance are as follows:

RENAMING SECTION

AREA SECTION

RECORD SECTION

SET SECTION

The RENAMING SECTION and the SET SECTION may be omitted if not required. The RENAMING SECTION consists of a single entry. The AREA, RECORD, and SET SECTIONS consist of an entry for each area, record, or set to be included in the subschema being defined.

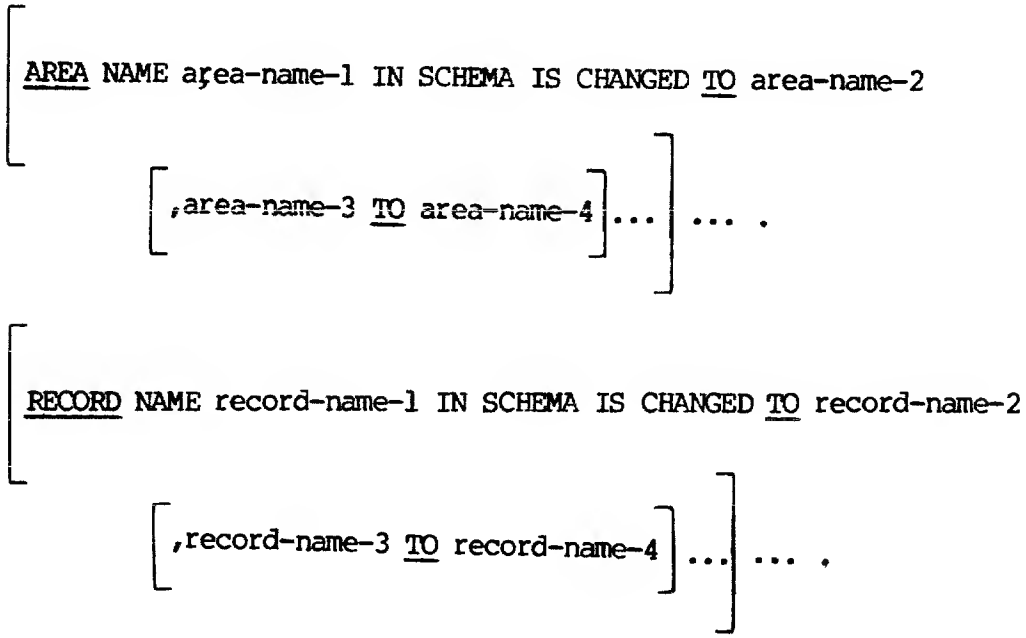
****
* RENAMING SECTION *
****

Function

To relate names in the subschema to names in the schema in order to achieve conformity with the naming conventions of COBOL or for convenience purposes.

Complete Entry

RENAMING SECTION



```
[ DATA NAME data-base-identifier-1 IN SCHEMA IS CHANGED TO data-base-data-name-1

        [ ,data-base-identifier-2 TO data-base-data-name-2 ]... ] ... .

[ SET NAME set-name-1 IN SCHEMA IS CHANGED TO set-name-2

        [ ,set-name-3 TO set-name-4 ]... ] ... .
```

Syntax Rules

1. In each renaming clause, the first name of each pair of names must have been previously declared in the schema; the second name of each pair must conform to COBOL naming rules. The second name is known as a synonym for description purposes.

2. The following categories of names declared in the schema can be the subject of RENAMING clauses in a subschema.

a. Area-names

b. Record-names

c. Set-names

d. Names of data aggregates or data items within each record type

e. Names of data items not within any record type

Within each category the collection of names for a subschema consists of:

a. The synonyms defined in RENAMING clauses for a particular subschema, plus

b. Those names declared in the schema which are known to the subschema and do not have synonyms in this subschema.

Each collection of names must not contain any duplicates. In addition, names of data items not within a record type cannot be the same as any record name.

3. Within each category, any name declared in the schema must not be the subject of more than one RENAMING clause in any one subschema.

4. If necessary to ensure uniqueness, database-identifier-1 and database-identifier-2 must be qualified, using the same record name declared in the schema.

General Rules

1. A name which is a synonym in a RENAMING clause must be used instead of the corresponding name declared in the schema throughout subsequent sections in the subschema and throughout any run-unit which invokes the subschema. This rule applies equally to names communicated by a run-unit to the DBMS and to names communicated by the DBMS to a run-unit.

2. Any name declared in a schema which is the subject of a RENAMING clause and appears in a subschema as result of a COPY area-name or COPY set-name clause, will be replaced by its corresponding synonym.

3. Whenever there is a requirement that an area, record, set, or data item be referred to by name in the subschema, then

- if a synonym exists as a result of a RENAMING clause in the subschema, this synonym must be used; failure to do this produces only a warning message and CSUBS replaces the schema name;

- otherwise, the name declared in the schema must be used.

4. All synonym names declared in the remaining section must be used once subsequently in this subschema. Failure to do this is a fatal subschema compiler error.

****
* AREA SECTION *
****

Function

To enumerate the areas of the schema that are included in the subschema and by implicaton, to remove from view all other areas of the schema.

Complete Entry

Format 1

COPY area-name-1 [,area-name-2] ... .

Format 2

COPY ALL AREAS.

****
* AREA SECTION-COPY AREA *
****

Function

To name all areas which are to be included in the subschema.

General Format

Format 1

COPY area-name-1 [, area-name-2]... ... .

Format 2

COPY ALL AREAS.

Syntax Rules

1. Area-name-1, area-name-2 must refer to areas defined in the schema.
2. Format 1 may be repeated as required.
3. If Format 2 is used, Format 1 entries are not allowed.

General Rules

1. Format 1 causes the entries for the referenced areas in the schema to be included in the subschema.
2. Format 2 causes all areas for which entries are included in the schema to be included in the subschema.

****
* RECORD SECTION *
****

Function

To name and define the records and subordinate data items within records of the schema that are to be included in the subschema.

By implication, to remove from view all other records and data items within records of the schema.

Complete Record Description Entry Skeleton

RECORD SECTION

Record Control Entry.

[Data Description Entry...].

Complete Record Control Entry

Format 1

01 record-name-1.

[;WITHIN area-name-1 [,area-name-2] ...] .

Format 2

77 data-base-data-name-2.

Complete Data Description Entry

```
level-number  data-base-data-name-1

[ ; { PICTURE } IS character-string ].
    { PIC     }

[ ; USAGE IS { COMPUTATIONAL   } ].
             { COMP            }
             { COMPUTATIONAL-3 }
             { COMP-3          }
             { DISPLAY         }
             { DATABASE-KEY    }

[ ; SIGN IS { LEADING  } SEPARATE CHARACTER ].
            { TRAILING }

[ ;OCCURS integer-2 TIMES                          ]
[ ;OCCURS integer-1 TO integer-2 TIMES             ].
[   DEPENDING ON data-base-data-name-3             ]
```

Syntax Rules for Complete Record Description Entry

1. A record control entry is required as the first element in a subschema record description; if Format 1, it may be followed by zero or more data description entries.

2. In a Format 1 record control entry, any WITHIN clause which follows the name of the record is optional and its order of appearance is immaterial.

3. Level-number in a data description entry may be any number from Ø2-49. (Format 1).

4. The PICTURE clause must be specified for every elementary item, except any data item defined with a USAGE IS COMPUTATIONAL or USAGE IS DATABASE KEY clause.

5. In data description entries, the clauses may be written in any order, with the exception that the data-base-data-name clause must immediately follow the level-number.

General Rules for Complete Record Description Entry

1. A record description entry extends from a record control entry to the next record control entry, or to the end of the RECORD SECTION.

2. A data description entry is used to describe group and elementary items associated with a Format-1 record control entry.

3. If the record referenced by record-name-1 has been defined in the schema as having no data, then only clauses included in the record control entry are permitted in this record description.

4. The PICTURE clause must not be specified, except for elementary data items. (see Syntax Rule 4).

```
******************************
* RECORD SECTION-RECORD NAME *
******************************
```

Function

To define records which are to be included in the subschema.

General Format

```
record-name.
```

Syntax Rules

1. All record-names must be unique within the record-names used in the subschema.
2. Record-name must refer to a record declared in the schema.

General Rules

None.

****
* RECORD SECTION-WITHIN CLAUSE *
****

Function

To define and restrict the selection of occurrences of the record named.

General Format

WITHIN area-name-1 [,area-name-2]... .

NOTE

At present, only syntax checking of this clause is done, and that is only partial: area-name-1, area-name-2 must refer to areas defined in the schema.

Syntax Rules

None.

General Rules

None.

```
****************************************
* RECORD SECTION  DATA-BASE-DATA-NAME *
****************************************
```

Function

To select data items and data aggregates of interest from a record defined in the schema, thereby implicitly removing from view all unnamed items of the record.

General Format

data-base-data-name .

Syntax Rules

1. Data-base-data-name may refer to any data item or data aggregate declared for the record in the schema.

2. Any elementary data item declared for this record in the subschema must have been declared in the schema as a data item.

3. Any data item or vector which is declared as a component of a repeating group in the schema can only be declared in the subschema as a component that is an elementary data item or an array, of that repeating group.

4. Data-base-data-names must be unique within the subschema record description entry.

General Rules

1. If data-base-data-name is the name of a repeating group declared in the schema for this record, any component of the group can be omitted from the subschema record description entry. If a subordinate repeating group is omitted, all its components are omitted.

****
* RECORD SECTION-LEVEL NUMBER *
****

Function

To show the hierarchy of data names within a record.

General Format

level-number .

Syntax Rules

1. A level-number is required as the first element in each data description entry.

2. Data description entries describe a record entry and may have level-numbers with values of Ø2 through 49.

General Rules

1. A level-number Ø1 entry identifies the first entry; that is, the record control entry.

2. Level-numbers assigned to data items and data aggregates in the schema record entry may be altered in the subschema declarations; however, for repeating groups, the relative levels within the group's hierarchy must be maintained in the subschema record.

****
* RECORD SECTION-OCCURS CLAUSE *
****

Function

To supply the information required for the application of subscripts or indices for repeated data, eliminating the need for separate entries.

To define arrays and repeating groups based on schema declarations of vectors and repeating groups.

General Format

Format 1

OCCURS integer-2 TIMES.

Format 2

OCCURS integer-1 TO integer-2 TIMES

DEPENDING ON data-base-data-name-1 .

Syntax Rules

1. Integer-1 and integer-2 must be unsigned nonzero integers. Where both are used, the value of integer-1 must be less than the value of integer-2.

2. The data description entry for data-base-data-name-1 must describe an integer.

3. Data-base-data-name-1 must not be the subject of an OCCURS clause, or subordinate to a group item containing an OCCURS clause.

4. Data-base-data-name-1 must refer to a data item defined in both the schema and subschema as belonging to the same record to which the item being described by this clause belongs. The description

of data-base- data-name-1 must precede the description of the data item described by the OCCURS clause in the total record description to which both items belong.

5. The OCCURS clause may only be specified in a data description entry that describes a data item that has been declared in the corresponding schema record entry as an element of a vector or a repeating group.

General Rules

1. The OCCURS clause is used in defining arrays and repeating groups. Whenever the OCCURS clause is used, the data-base-data-name which is the subject of this entry must be subscripted whenever it is referenced in a COBOL statement other than SEARCH. Further, if the subject of this entry is the name of a group item, then all data-base-data-names belonging to the group must be subscripted whenever they are used as operands.

2. Except for the OCCURS clause itself, all data description entry clauses associated with an item whose data description entry includes an OCCURS clause apply to each occurrence of the item described.

3. In Format 1, the value of integer-2 represents the exact number of occurrences; in Format 2, the value of integer-2 represents the maximum number of occurrences in the data-base and the exact number of occurrences within the COBOL program.

4. Format 2 specifies that the subject of this entry has a variable number of occurrences in the database only. This does not imply that the length of the subject is variable, but that the number of occurrences is variable.

5. The value of data-base-data-name-1 is the count of the number of occurrences of the subject and its value must not exceed integer-2 or be less than integer-1. Reducing the value of data-base-data-name-1 makes the contents of data items, whose occurrence numbers now exceed the value of data-base-data-name-1, unpredictable.

6. If Format 1 is used, the number of occurrences specified by integer-2 in the subschema must be less than or equal to the number of occurrences declared in the schema for the corresponding vector or repeating group. If Format 2 is used, data-base-data-name-1 must be identical to the depending variable name specified in the schema.

```
**********************************
* RECORD SECTION-(PICTURE CLAUSE) *
**********************************
```

Function

To describe the general characteristics of elementary items as they appear in the User Working Area.

General Format

```
[ ; { PICTURE } IS character-string ].
    { PIC     }
```

Syntax Rules

1. A PICTURE clause can be specified only at the elementary item level.
2. A character-string consists of certain allowable combinations of characters in the COBOL character set used as symbols.
3. The maximum number of symbols allowed in the character-string is 30.
4. The PICTURE clause describes fixed length data items.
5. PIC is an abbreviation for PICTURE.
6. The PICTURE clause must be specified for every elementary item except when USAGE is COMPUTATIONAL or DATABASE-KEY.

General Rules

1. There are three categories of data that can be described with a PICTURE clause: Alphabetic, Numeric and Alphanumeric.
2. To define an item as Alphabetic:
   a. Its PICTURE character-string can only contain the symbol 'A'; and

b. Its contents when represented in Standard Data Format must be any combination of the twenty-six (26) letters of the Roman alphabet and the space from the COBOL character set.

3. To define an item as Numeric:

   a. Its PICTURE character-string can only contain the symbols '9', 'P', 'S' and 'V'. The number of digit positions that can be described in the PICTURE character-string must range from 1 to 18 inclusive;

   b. If unsigned, its contents when represented in Standard Data Format must be a combination of the Arabic numerals 'Ø', '1', '2', '3', '4', '5', '6', '7', '8', and '9'; if signed, the item may also contain a '+', '-', or other representation of an operational sign.

4. To define an item as Alphanumeric:

   a. Its PICTURE character-string is restricted to certain combinations of the symbols 'A', 'X', '9', and the item is treated as a character-string containing all 'X's. (A PICTURE character-string which contain all 'A's or all '9's does not define an Alphanumeric item); and

   b. Its contents when represented in Standard Data Format are allowable characters in the computer's character set.

5. The size of an elementary item, where size means the number of character positions occupied by the elementary items in Standard Data Format, is determined by the number of allowable symbols that represent character positions. An unsigned nonzero integer which is enclosed in parentheses following the symbols 'A', 'X', '9', 'P', indicates the number of consecutive occurrences of the symbol. Note that the symbols 'S' and 'V' may appear only once in a given PICTURE.

6. The function of the symbols used to describe an elementary item are explained as follows:

   A Each 'A' in the character-string represents a character position which can contain only a letter of the alphabet or a space.

   P The 'P' indicates an assumed decimal scaling position and is used to specify the location of an assumed decimal point, when the point is not within the number that appears in the data item. The scaling position character 'P' is not counted in the size of the data item. Scaling position characters are counted in determining the maximum number of digit positions (18) in numeric items which appear as operands in arithmetic statements. The scaling position character 'P' can appear only to the left or right as a continuous string

of 'P's within a PICTURE description; since the scaling position character 'P' implies an assumed decimal point (to the left of 'P', if 'P's are the left-most PICTURE characters and to the right of 'P's if 'P's are the right-most PICTURE characters), the assumed decimal point symbol 'V' is redundant as either the left-most or right-most character within such a PICTURE description.

S The letter 'S' is used in a character-string to indicate the presence, but neither the representation nor, necessarily, the position of an operational sign; it must be written as the left-most character in the PICTURE. The 'S' is not counted in determining the size (in terms of Standard Data Format characters) of the elementary item unless the entry is subject to a SIGN clause which specified the optional SEPARATE CHARACTER phrase.

V The 'V' is used in a character-string to indicate the location of the assumed decimal point and may only appear once in a character-string. The 'V' does not represent a character position and therefore is not counted in the size of the elementary item. When the assumed decimal point is to the right of the right-most symbol in the string, the 'V' is redundant.

X Each 'X' in the character-string is used to represent a character position which contains any allowable character from the computer's character set.

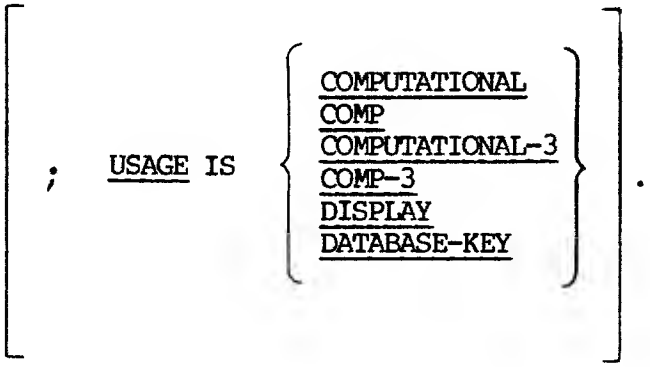
9 Each '9' in the character-string represents a character position which contains a numeral and is counted in the size of the item.

7. The Subschema PICTURE clause describes the format and characteristics of schema data items as they appear in the User Working Area. The characteristics of data items defined in the subschema may differ from the characteristics of the corresponding data items defined in the schema. Variations are limited to those differences which would not violate the rules stated in the subschema for the OCCURS clause, USAGE clause, etc. When characteristics differ, conversion rules are employed in transferring data to and from the User Working Area.

```
********************************
* RECORDS SECTION-USAGE CLAUSE *
********************************
```

Function

To specify the format of the COBOL data items and to cause certain conversions between an item as it is defined in the schema and as it is desired in the User Working Area.



Syntax Rules

1. The PICTURE of a COMPUTATIONAL-3 item can contain only '9', the operational sign character 'S', the implied decimal point character 'V', one or more 'P's.

2. The USAGE IS DISPLAY clause indicates that the format of the data is Standard Data Format.

3. COMP is an abbreviation for COMPUTATIONAL-3.

4. The USAGE IS DATABASE-KEY clause defines a data item designed to hold a database-key.

5. The format of a data item described with a USAGE IS DATABASE-KEY clause is PICTUREX(6); no other clauses, may be specified for the data item.

6. The clause USAGE IS DATABASE-KEY may be used in the subschema if and only if it describes an item declared in the schema as a database-key.

General Rules

1. The USAGE clause can be written at any level. If the USAGE clause is written at a group level, it applies to each elementary item in the group. The USAGE clause of an elementary item cannot contradict the USAGE clause of a group to which the item belongs.

2. The USAGE clause specifies the manner in which a data item is represented in the User Working Area and may affect the radix or type of character representation of the item.

3. COMPUTATIONAL-3 items are represented as packed decimal data.

4. If the USAGE clause is not specified for an elementary data item or for any group item to which it belongs, the USAGE is assumed to be DISPLAY.

5. If the USAGE IS DISPLAY clause is combined with a PICTURE clause that specifies a numeric category, the data item is represented as unpacked decimal data.

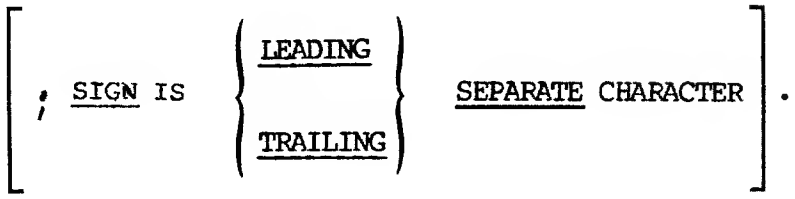
6. When USAGE is COMPUTATIONAL or DATABASE-KEY, the use of the PICTURE clause is prohibited.

```
*******************************
* RECORD SECTION-SIGN CLAUSE *
*******************************
```

Function

The SIGN clause specifies the position and the mode of the representation of the operational sign.

General Format



Syntax Rules

NOTE

To be implemented in Rev. 15 or a subsequent release of the DBMS.

1. The SIGN clause may be specified only for a numeric Data Description entry whose PICTURE contains the character 'S' or a group item containing at least one such numeric Data Description entry.

2. The numeric Data Description entries to which the SIGN clause applies must be described as USAGE IS DISPLAY.

3. At most, one SIGN may apply to any given numeric Data Description entry.

General Rules

1. The optional SIGN clause, if present, specifies the position and the mode of representation of the operational sign for the numeric Data Description entry to which it applies.

   The SIGN clause applies only to numeric Data Description entries whose Subschema PICTURE contains the character 'S' (the 'S' indicates the presence of, but neither the representation nor, necessarily, the position of the operational sign).

2. A numeric Data Description entry whose PICTURE contains the character 'S', but to which no optional SIGN clause applies, has an operational sign, but neither the representation nor, necessarily, the position of the operational sign is specified by the character 'S'. In this (default) case, the position and representation of the operational sign will be embedded trailing.

3. If the optional SEPARATE CHARACTER phrase is not present, then:

   a. the operational sign will be associated with the leading (or, respectively, trailing) digit position of the elementary numeric data item.

   b. the letter 'S' in a PICTURE character-string is not counted in determining the size of the item (in terms of Standard Data Format characters).

4. If the optional SEPARATE CHARACTER phrase is present:

   a. the operational sign will be presumed to be the leading (or, respectively, trailing) character position of the elementary numeric data item; this character position is not a digit position.

   b. the letter 'S' in a PICTURE character-string is counted in determining the size of the item (in terms of the Standard Data Format characters).

   c. the operational sign for positive and negative are the Standard Data Format characters '+' and '-', respectively.

****
* DATABASE DATA NAME *
****

Function

To select data-base-data-names which appear in the schema, but not in any data sub-entry of a record.

General Format

77 data-base-data-name-2.

Syntax Rules

1. Data-base-data-name-2 may raefer to data-base-data-names defined in the Schema as data-base-keys used in LOCATION MODE IS DIRECT, AREA-ID IS, and ALIAS within the set selection clause.

2. The format of data-base-data-name-2 will be the same as defined inthe Schema and cannot be modified by the subschema. Therefore, no usage or PICTURE clause is allowed.

General Rules

None

****
* SET SECTION *
****

Function

To name and define the sets of the schema that are to be included in the subschema and by implication, to remove from view all other sets of the schema;

Complete Entry

SET SECTION

Format 1

COPY set-name-1 [,set-name-2] ... .

Format 2

COPY ALL SETS.

```
************************
* SET SECTION-COPY SET *
************************
```

Function

To name all sets which are to be included in the subschema.

General Format

Format 1

COPY set-name-1 [,set-name-2] ... .

Format 2

COPY ALL SETS .

General Rules

1. Format 1 causes the entries for the referenced sets in the schema to be included in the subschema. No changes are allowed.

2. Format 2 causes all sets for which entries are included in the schema to be included in the subschema.

3. A record description entry for the owner record of each set included in a subschema, must also be included in that subschema.

SECTION 3

COBOL DATA MANIPULATION LANGUAGE PROCESSOR (CDML)

OVERVIEW

The purpose of the COBOL Data Manipulation Language (CDML) is to support access to the Data Manipulation Command Processor (DMLCP) using COBOL as a host language. The CDML language consists of English-like statements and is based on the April 71 report from the CODASYL DBTG. These statements are translated by the CDML preprocessor into COBOL Working Storage data-descriptions, COBOL call statements, and other COBOL procedure statements.

Before a CDML program can be written, a COBOL SUBSCHEMA must be written and compiled by the COBOL SUBSCHEMA compiler (FSUBS) described in Section 2 of this manual. The COBOL SUBSCHEMA contains the names and data-descriptions of the areas, sets, items, records and data-base-data-names that the CDML application program can reference.

HOW TO USE THE CDML PREPROCESSOR

Programming Tips

Once the subschema has been provided, the following programming tips should be considered when writing application programs.

1. All COBOL program units which contain CDML commands must contain a SUBSCHEMA statement.

2. Only one subschema of one schema can be invoked at any one time in a DBMS application program.

3. The application program and subprograms which contain DML commands must be programmed in one language, e.g., all COBOL or all FORTRAN.

4. The COBOL DML application program must reference a COBOL SUBSCHEMA.

5. The COBOL application program must avoid using the USER WORK AREA register names which are inserted into Working Storage by the COBOL SUBSCHEMA compiler (see page 3-99), and any names beginning with the characters "DML".

6. The procedure section of the COBOL application program may freely intermix COBOL statements and CDML statements; however, CDML statements and COBOL statements may not be mixed on the same line or on a continuation line.

7. Only one INVOKE statement should be executed during the duration of a run-unit. The execution of more than one INVOKE has undefined results. The failure to execute an INVOKE statement will cause any CDML statements to be ignored.

8. All CDML statements must be preceded by a '#' to identify the statement to the CDML preprocessor.

9. All CDML programs must terminate execution with an EXIT DBMS statement.

CDML ORGANIZATION

Two Classes of CDML Statements

There are two general classes of statements in the CDML, declaration statements and executable commands.

The declaration statements allow the CDML program to declare the SCHEMA and the SUBSCHEMA required for the application program. It also inserts the definition of a part of COBOL which serves as the User Work Area (UWA) for the application program. The USER WORK AREA contains the items and their PICTURE declared by the subschema and the DBMS registers.

The CDML executable commands specify the individual DML processes required by the CDML application program. Currently the following CDML capabilities exist:

Run-Unit Oriented Commands

Run-Unit Oriented Commands are used by the run-unit to initiate and terminate communication with the DBMS. They also allow the user to clear error statuses and declare Privacy keys. They include: INVOKE, EXIT DBMS, ABORT DBMS, CLEAR ERROR and PRIVACY KEY.

Area Oriented Commands

Area Oriented Commands obtain and relinquish control of the DBMS area files. They also allow the user to specify the mode in which these files are to be used. They include: OPEN and CLOSE.

Record Oriented Commands

Record Oriented Commands are used to create, manipulate, remove and obtain records from the database. These commands include: STORE, DELETE, GET, MODIFY, FIND and FETCH.

Set Oriented Commands

Set Oriented Commands manipulate set membership. They include: INSERT, REMOVE and MODIFY.

Supporting Commands

Supporting Commands allow the user to manipulate currency statuses. They include: SUPPRESS, MOVE, IF and CLEAR.

Concurrency Functions

Concurrency Functions allow a run-unit to access and update a database along with concurrent run-units. They insure that the view of the database is consistent, guarantee its integrity and provide for rollback and recovery facilities. These commands include: START TRANSACTION, END TRANSACTION and ABORT TRANSACTION.

CDML SYNTAX COMPONENTS AND NOTATION

This section describes the requirements for the COBOL Data Manipulation Language (CDML) statements and syntax notation used in programming COBOL Application Programs to run on Prime's DBMS system. The first subsection describes the syntax components which may appear in an CDML statement. The next subsection describes the syntax notation which will be used in defining the permissible syntax within the statements.

CDML Statement Format Rules

The CDML statements are in an English-like format but they have some of the characteristics of a COBOL statement. They must start in column 7 and end before column 73. A statement may be continued simply by omitting the terminating period and putting a '#' character in column 7 of the next statement. There is no limit to the number of continuation statements that may be used. Note that all CDML statements and declarations must start with a "#" character in column 7 to be recognized by the CDML preprocessor.

Character Set

The CDML statements are composed of tokens which are themselves composed from the character sets shown in Table 3-1.

Terminating CDML Statements

CDML statements must be terminated by a period.

Table 3-1. CDML STATEMENTS CHARACTER SET

CHARACTER GROUP	MEMBERS
Letters	A B C D E F G H I
	J K L M N O P Q R S
	T U V W X Y Z
DIGITS	1 2 3 4 5 6 7 8 9 0
Special Characters	blank (space)
	- (minus sign or dash)
	, (comma)
	; (semicolon)
	. (period)
	' (quotation mark)
Reserved for future use	
	= (equal sign)
	> (greater than sign)
	< (less than sign)
	* (Asterisk)
	$ (Dollar sign)

Delimiting Characters

The following CDML characters serve to delimit syntax components: 1) space, 2) comma, 3) semicolon and 4) period. Consecutive spaces are treated as a single space. The end of a line also serves as a delimiting character so that a syntax component may not cross a line boundary.

Generic Terms

The following is a list of generic terms for the CDML Language and the rules for their function.

Schema-name

Used to identify that schema the run-unit is to access. It must identify a valid schema compiled by the DDL.

Subschema-name

Used to identify the subschema which the run-unit is to access. It must identify a COBOL subschema which has been compiled by CSUBS.

COBOL-label

Refers to a COBOL paragraph name or section name which can be accessed by a GO TO.

Integer

Used in CDML statements to provide a literal integer value. The integer syntax component obeys the normal COBOL rules for a COMPUTATIONAL item.

Literal

Used in the CDML statements to refer to a literal value that is to be passed for a PRIVACY KEY statement. The literal component must refer to a valid COBOL integer, decimal or display constant.

Identifier

Used in CDML statements to refer to a COBOL variable name.

Data-base-data-name (dbdn)

Refers to a storage location which has been declared by the schema and the subschema to hold a particular value such as an area name or data base-key.

Data-base-identifier (dbid)

Refers to a storage location which has been declared by the schema and the subschema to hold the value of an item of a record. The dbid is a valid COBOL identifier and is declared in the COBOL application program in the User Work Area in Working Storage.

Area-name

Refers to a valid area name which has been defined by the schema. It is formed by providing the identifier which has been defined in the invoked subschema.

Record-name

Refers to the database record type which has been declared in the schema. It is formed by providing the identifier which has been defined in the invoked subschema.

Set-name

Refers to the database set type which has been declared in the schema. It is formed by providing the identifier which has been defined in the invoked subschema.

CDML Syntax Notation

This subsection defines the conventions used to describe syntactically correct CDML statements.

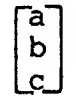
A syntax skeleton consists of a sequence of syntax component names, delimiters, and keywords, which may be grouped by special notational symbols. The skeleton represents a source specification of input to CDML in which the sequence of actual syntax components, delimiters, and keywords corresponds to that in the skeleton. The special notational symbols denote whether optional, alternative, or repeated sequences may occur within the corresponding source specification.

The following conventions apply within a syntax skeleton:

1. All underlined upper-case words are required when the format is used.

2. Upper-case words which are not underlined are optional words and need not be used.

3. Lower-case words are generic terms which must be replaced by appropriate names or values.



A source specification is not required in this position but it may contain either a, b, or c.

A source specification in this position must include exactly one of the components a, b, or c.

A source specification in this position may consist of a list of one or more of a, b, or c. If more than one is specified, they must be separated by commas.

... The immediately preceding construct may be repeated an arbitrary number of times.

CDML PREPROCESSOR COMMANDS

This subsection describes in detail the CDML declarations and commands. The description of each clause contains the following headings:

Function

A brief narrative description of the function of the facility.

General Format

The arrangement of the syntax elements which make up the clause.

Syntax Rules

These serve to amplify or restrict the usage of the elements within the general format.

General Rules

1. A description of the semantic rules for using the command and a description of program capability and error conditions which relate to the statement.

Examples and Discussion

Specific examples of the use of the statement.

COMPLETE SYNTAX SKELETON

```
ABORT TRANSACTION identifier-1 .

CLEAR ERROR .

CLOSE

    Format 1

        CLOSE ALL AREAS .

    Format 2

        CLOSE AREA[S] area-name-1 [,area-name-2]... .

DELETE [ MANDATORY ]
       [ SELECTIVE ] .
       [ ALL       ]

END TRANSACTION identifier-1 .

EXIT DBMS .
```

```
{FETCH}
{FIND } rse .
```

Record Selection Expressions

Format 1

```
USING identifier-1 .
```

Format 2

```
[{OWNER  } IN set-name OF] CURRENT OF {RECORD record-name-2 }
 {MEMBER }                            {SET set-name-4       }
                                      {AREA area-name-1     } .
                                      {RUN-UNIT             }
```

Format 3

```
{NEXT      }
{PRIOR     }
{FIRST     } RECORD [record name-3] OF {SET set-name-5 } .
{LAST      }                           {AREA area-name-2}
{integer-1 }
{identifier-2}
```

Format 4

```
[NEXT DUPLICATE WITHIN] RECORD record-name-4 .
```

Format 5

```
record-name-5 VIA [CURRENT OF] SET set-name-7 [USING dbid-3 [,dbid4]...] .
```

Format 6

```
NEXT DUPLICATE WITHIN SET set-name-8 USING dbid-5 [,dbid-6]... .
```

GET [dbid-1 , dbid-2...] .

IF

Format 1

```
IF set-name-1 SET [NOT] EMPTY { cobol-procedure-name-1 } [ ELSE cobol-procedure-name ] .
                              { NEXT                   }
```

Format 2

```
IF RECORD [NOT] { MEMBER } OF { set-name-2 } { cobol-procedure-name-3 }
                { OWNER  }    { ANY set    } { NEXT                  }

    [ELSE cobol-procedure-name-4].
```

```
INSERT INTO { SETS set-name-1 [,set-name2]... } .
           { ALL SETS                         }
```

INVOKE DBMS .

MODIFY [dbid-1 , dbid-2...].

MOVE

Format 1

```
MOVE CURRENCY STATUS FOR { RUN-UNIT
                         { RECORD record-name
                         { AREA area-name
                         { SET set-name          } TO identifier-1 .
```

Format 2

```
MOVE { RECORD-NAME } FOR { RUN-UNIT
     { AREA-NAME   }     { RECORD record-name
                         { AREA area-name
                         { SET set-name
                         { identifier-2          } TO identifier-3 .
```

ON ERROR

Format 1

;ON ALL ERRORS GO TO cobol-procedure-name .

Format 2

;ON ERROR integer-1 [,integer-2]... GO TO cobol-procedure-name]... .

[ON ERROR integer-3[,integer-4] ... GO TO cobol-procedure-name]... .

[ON OTHER ERRORS GO TO cobol-procedure-name].

OPEN

Format 1

```
OPEN ALL AREAS [ USAGE-MODE IS [ EXCLUSIVE ] { RETRIEVAL } ] .
               [ PROTECTED ]  { UPDATE    }
```

Format 2

```
OPEN AREA[S] area-name-1 [,area-name-2]...
    [ USAGE MODE IS [ EXCLUSIVE ] { RETRIEVAL } ] .
                   [ PROTECTED ]  { UPDATE    }
```

PRIVACY KEY

Format 1

```
PRIVACY KEY [FOR [ [EXCLUSIVE] RETRIEVAL ] ]OF
             [ [PROTECTED]                ]
             [ [EXCLUSIVE] UPDATE         ]
             [ [PROTECTED]                ]
  { AREAS area-name-1 [,area-name-2]... } IS { literal-1    } .
  { ALL AREAS                           }    { identifier-1 }
```

Format 2

```
PRIVACY KEY [FOR { REST                  } ] OF
                 { STORE                 }
                 { GET                   }
                 { MODIFY                }
                 { INSERT                }
                 { REMOVE                }
                 { DELETE MANDATORY      }
                 { DELETE SELECTIVE      }
                 { DELETE ALL            }
                 { FIND                  }
  { RECORDS record-name-1 [,record-name-2]... } IS { literal-2    } .
  { ALL RECORDS                               }    { identifier-2 }
```

Format 3

```
PRIVACY KEY [FOR { REST | STORE | GET | MODIFY } ]OF DATA-ITEMS dbid-1 [,dbid-2]...

                IS { literal-3 | identifier-3 } .
```

Format 4

```
PRIVACY KEY [FOR { REST | INSERT | REMOVE | FIND } ] OF { SETS set-name-1 [,set-name-2]... | ALL SETS }

                IS { literal-4 | identifier-4 } .
```

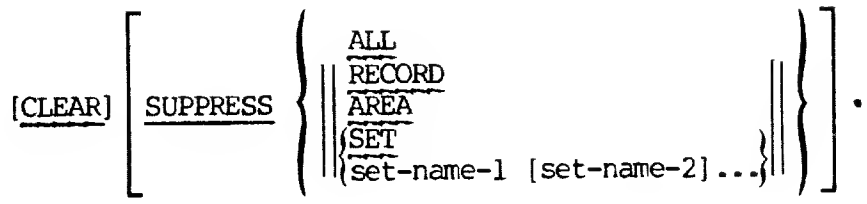
```
REMOVE FROM { SETS set-name-1 [,set-name-2]... | ALL SETS } .
```

```
START TRANSACTION identifier [ UPDATE | ,RETRIEVAL ] .
```

```
STORE record-name .
```

SUBSCHEMA subschema-name of SCHEMA schema-name .

SUPPRESS



DML STATEMENTS

****
* ABORT TRANSACTION *
****

Function

Causes all update actions since the last START OF TRANSACTION command to be negated (i.e., the database is rolled back) ; all locks to be released and all currency indicators to be reset as they were at the last START TRANSACTION command.

General Format

ABORT TRANSACTION identifier-1.

Syntax Rules

1. Identifier-1 must contain the value assigned by the START OF TRANSACTION command.
2. Identifier-1 must be a level 77 name with USAGE COMPUTATIONAL.

General Rules

1. If before-imaging is turned off for the INVOKED schema (using DBACP) then the ABORT TRANSACTION has no effect.
2. If identifier-1 is invalid or no transaction is currently active, error status condition 2535 results and CONTYP will contain more detailed error information.

Examples and Discussion

See START TRANSACTION.

****
* CLEAR ERROR *
****

Function

CLEAR ERROR clears DBMS error conditions and the associated registers.

General Format

CLEAR ERROR.

Syntax Rules

None

General Rules

1. This command clears all DBMS registers in the User Work Area, including ERSTAT, CONTYP, ERAREA, ERDEC, ERSET, ERITEM, ERCASE.
2. If the error is a non-fatal error, it clears an internal flag allowing the DBMS to continue processing.
3. It must be called after every DBMS error condition.
4. No Error Status Codes can result from CLEAR ERROR.

Examples and Discussion

See ON ERROR clause.

****
* CLOSE *
****

Function

To relinquish control over the specified areas and make them available to other run-units.

General Format

Format 1

CLOSE ALL AREAS.

Format 2

CLOSE AREA[S] area-name-1 [,area-name-2]... .

Syntax Rules

1. All area names specified must be included in the invoked schema.

General Rules

1. Note that CLOSE (unlike OPEN) need not deal with all opened areas.

2. After execution of a CLOSE statement for a given area, any attempt to access that AREA will result in Error Status Condition nnØ1 (where nn indicates the particular DML statement attempted). A subsequent CLOSE statement does not return an Error Status- the command is treated as a no-op.

3. In format 1, the CLOSE statement applies to all areas which are in an open status for the run-unit. These areas are then subject to General Rule 2 above.

4. The run-unit's currency indicators may identify those record occurrences located in areas which are no longer in an open status for that run-unit. These currency indicators become null after the CLOSE statement is executed.

5. If all area-names specified are not included in the invoked subschema, Error Status Condition Ø146 results; otherwise ERSTAT is zero, indicating successful execution of the CLOSE statement.

6. A CLOSE statement cannot be executed during an open transaction, otherwise Error Status Condition Ø135 will result with CONTYP set to 15.

Examples and Discussion for the CLOSE Statement

Example 1

```
#   CLOSE AREAS MY-AREA, YOUR-AREA.
```

This statement closes two areas: 1) MY-AREA and 2) YOUR-AREA. All records in these areas are no longer available for processing and all of the currency statuses that refer to records in those areas are cleared.

****
* DELETE *
****

Function

The DELETE statement performs the following functions:

1. Makes the object record occurrence unavailable for further processing by the imperative-statements of the DML.

2. Removes the object record from all set occurrences in which it is a member.

3. Deletes all record occurrences which are mandatory members of set occurrences owned by the object record.

4. Removes or deletes optionally all record occurrences which are optional members of set occurrences owned by the object record.

5. Optionally prevents deletion of the object record if the database contains any non-empty set occurrences of which the object record is the owner.

General Format

```
DELETE [ MANDATORY ]
       [ SELECTIVE ]
       [ ALL       ] .
```

Syntax Rules

none

General Rules

1. The immediate object record occurrence of the DELETE statement is the current record of the run-unit.

2. The object record is removed from all set occurrences in which it is a member. It is then deleted; that is, made unavailable for further processing by any DML statement.

Note that there is a distinction made between a removed record occurrence and a deleted record occurrence. A DELETE statement is inclusive of the function of a REMOVE statement, and is described in the previous paragraph of this general rule. A REMOVE statement cancels the existing membership of a record occurrence in specific set occurrences. The removed record is then not accessible through those set occurrences but continues to be accessible through any other sets in which it participates as a member. It may also be accessible by virtue of its having been defined with a LOCATION MODE IS CALC clause. It is always accessible by means of a complete scan of the area in which it participates, or through its database-key (if that is known).

3. The unqualified form of the DELETE statement deletes the object record only if it has no member records. If the object record is the owner of a non-empty set occurrence, the DELETE statement is not successfully executed and Error Status Condition Ø23Ø results.

4. The DELETE MANDATORY form of the statement deletes the object record and all of its mandatory members. It removes but does not delete its optional members.

   If any deleted mandatory member is the owner of a set occurrence, then the DELETE statement is executed on such records as if it were the object record of a DELETE MANDATORY statement. Thus, all mandatory members of such sets are also deleted, which in turn causes this process to continue down the hierarchy.

5. The DELETE SELECTIVE form of the statement has the same results as the DELETE MANDATORY statement with the exception that:

   - Optional members are deleted if they do not currently participate as members in other set occurrences.

   - All deleted records which are themselves the owners of any set occurrences are treated as if they were the object of a DELETE SELECTIVE statement.

6. The DELETE ALL form of the statement deletes the object record together with all of its member records, regardless of whether they are mandatory or optional. As with the DELETE MANDATORY form of the statement, the delete process continues down the hierarchy with the difference that all deleted records, which are themselves the owners of any set occurrences, are treated as if they were the object record of a DELETE ALL statement.

7. The current record of the run-unit becomes null. No other currency status information is altered. Thus, the object record and all other records deleted or removed remain as current of area-name, record-name, and of all set-names in which they were current prior to the execution of the DELETE statement.

8. If the run-unit has not satisfied the privacy locks for the object record itself, or, for any other record occurrence which would be deleted, removed, or modified as a result of the execution of the DELETE statement, Error Status Condition 0204F will result and the run-unit will be aborted by DBMS.

9. If the areas in which the records are being deleted are opened for concurrent update and this command causes a concurrent update conflict or no update transaction is currently active, then error 0235 will result with concurrent error status found in CONTYP.

10. In addition to the conditions described in General Rule 8, any of the following conditions will result in an error if they occur during the execution of a DELETE statement:

    - If the current record of run-unit is not known, Error Status Condition 0213 results.

    - If the unqualified form of the DELETE statement is attempted against the owner record of a non-empty set occurrence, Error Status Condition 0230 results.

    - If any of the record occurrences which would be deleted, removed, or modified as a result of the execution of the DELETE statement is in an area which is not open, Error Status Condition 0201 results.

    - If the object record, or any record that would be deleted or removed as a result of executing the DELETE statement, is located within an area that is open for RETRIEVAL, Error Status Condition 0209 results.

    - If any record occurrence needed by the DBMS for informational purposes (such as following a search path) is not available because it is off-line or under exclusive control of another run-unit, Error Status Condition 0218 results.

    - The subschema invoked must name:

      a. All of the records which would be deleted, removed or modified as a result of executing the DELETE statement.

      b. All of the sets in which any record to be deleted is an owner or a mandatory member,

      c. All of the sets from which any record is to be removed.

      d. All of the owner record types of the sets from which records are being removed. Otherwise, Error Status Condition 0208 results.

11. When an error occurs, the database remains in the state existing prior to the attempted execution of the DELETE statement and the appropriate Error Status Condition code is made available in special register ERSTAT. Otherwise, ERSTAT is set to zero, indicating successful execution of the DELETE statement.

Error Status Codes For DELETE Statement

Condition	Content of Error Status
Area not open	0201
Referenced record-name or set-name not in invoked sub-schema	0208
Incorrect usage mode for area	0209
No current record of run-unit	0213
Implicitly referenced area not available	0218
Concurrent update conflict	0235
Unqualified DELETE command attempted on non-empty set	0230
Fatal - privacy breach attempted	-0204(0204F)

Examples and Discussion for the DELETE Statement

Example 1

```
# DELETE ALL.
```

This statement will delete the record that is the current record of run-unit and all of its members and remove it from all sets in which it is a member. It will then repeat this process moving down the hierarchy until all of the records which participate in the hierarchy which has the current record of run-unit as its root node have been deleted.

Example 2

```
# DELETE.
```

This statement will delete the current record at run-unit only if it owns no sets or if the sets which it does own currently have no members. It also removes the record from all sets in which it participates as a member.

****
* END TRANSACTION *
****

Function

END TRANSACTION defines the end of a recoverable program sequence and causes a record of the successful termination to be written to the DBMS system LOG file.

The DBMS guarantees that all updates are secure to disk.

The end of an update transaction allows modified data to be accessed by other transactions by releasing all locks.

General Format

END TRANSACTION identifier-1.

Syntax Rules

1. Identifier-1 is a level 77 COBOL identifier with USAGE COMPUTATIONAL which contains the value assigned by the most recent START TRANSACTION command to the current transaction.

General Rules

1. If Logging is turned off, then the END TRANSACTION has no effect.

2. If Identifier-1 does not match the identifier on the most recent START TRANSACTION, or no transaction is currently active, Error Status Condition 2435 results and CONTYP contains more detailed information.

Examples and Discussion

(See START TRANSACTION)

****
* EXIT DBMS *
****

Function

To terminate communication with the DBMS. To close all opened files and clear all status registers.

General Format

EXIT DBMS.

Syntax Rules

None

General Rules

1. An EXIT DBMS must be executed before termination of a run-unit.
2. Once an EXIT DBMS has been executed, the only way communication to the DBMS can be reinitialized is through an INVOKE.
3. An EXIT DBMS cannot occur inside an opened DBMS transaction, otherwise, Error Status Condition 1035 results with CONTYP set to 15.

Examples and Discussion

NONE

****
* FETCH *
****

Function

The FETCH statement is a combined FIND and GET statement which both establishes currency for the object record and makes available the contents of that record in the User Work Area.

General Format

FETCH rse.

Syntax Rule

See Record Selection Expressions.

General Rules

1. Execution of a FETCH statement causes the affected record to become available to the program; that record becomes the current record of the run-unit. A FETCH command is logically identical to execution of a FIND followed by a GET of all data items in that record.

2. Refer to the discussion of the FIND statement.

3. Refer to the discussion of the GET statement.

Error Status Codes for the FETCH Statement

Condition	Content of ERROR-STATUS
Area not open	2201
Database-key inconsistent with area-name	2202
Data items invalid or inconsistent	2204
Current record of set, area, record-name not known	2206
End of set or area or DUPs	2207
No current record of run-unit	2213
Implicitly referenced area not available	2218
Conversion of value of data item not possible	2219
Record not currently a member of named set	2222
Illegal area-name	2223
No record satisfies the FIND specified	2226
Concurrent access error	2235
Attempted to find owner of a singular set	2233
Record type not a member of named set	2240
Record type not owner of named set	2241
Record type not included in named area	2242
Arguments of Location Mode Clause not included in subschema	2243
Arguments of Set Occurrence Selection Clause not included in subschema	2244
Location mode of record not specified as CALC	2245
Specified data item, record, set or area not in subschema	2246
Fatal - privacy breach attempted	-2204(2204F)

****
* FIND *
****

Function

The FIND statement specifies a record occurrence as:

1. The current record of the run-unit.
2. The current record of the area in which it is stored.
3. The current record of its record-name.
4. The current record of set for all set-names in which it currently participates as owner or member.

A prior SUPPRESS may prevent the establishment of the object record occurrence as:

1. The current record of the area in which it is stored.
2. The current record of its record-name.
3. The current record of set for all, or specified sets in which it currently participates as an owner or member.

General Format

FIND rse.

NOTE:

Refer to rse for a discussion of the record-selection-expression.

Syntax Rules

See Record Selection Expression.

General Rules

1. Execution of a FIND statement causes the record referenced by the record-selection-expression to become the current record of the run-unit.

2. Execution of a FIND statement does not make the selected record available to the program - it merely identifies the record for use in certain subsequent statements. These include: GET, MODIFY, INSERT, REMOVE and DELETE.

3. If the suppression of updating currency indicators has not been set, the object record also becomes the current record of its area-name, the current record of its record-name, and the current record of all set-names in which it is defined as an owner or in which it currently participates as a member.

4. The effect of a SUPPRESS command or a series of SUPPRESS commands is to specify the area, record, set, and set-name currency indicators which are to retain their existing status.

5. If any of the following conditions are encountered, the FIND statement is not successfully executed, the database remains in the state existing prior to the attempted execution, and the appropriate Error Status Condition code is made available in special register ERSTAT. Otherwise, ERSTAT is set to zero, indicating successful execution of the FIND statement.

   - If the sought record is in an area which has not been opened, Error Status Condition Ø3Ø1 results.

   - If any record occurrence along the search path of the FIND statement is in areas which are off-line or under the exclusive control of a concurrent run-unit, Error Status Condition Ø318 results.

   - If any current record of the type specified is not known, Error Status Condition Ø3Ø6 results.

   - If a database-key is supplied or developed which is incompatible with the areas specified, Error Status Condition Ø3Ø2 results.

   - If an end-of-set, end-of-area, or end-of-Dups condition is encountered, Error Status Condition Ø3Ø7 results.

   - If no record in the area satisfies the FIND specified, Error Status Condition Ø326 results.

   - If a specific record type is sought in a set occurrence or is used in a FIND to define the current set occurrence of a set, and the record type is not defined as a member of the set in the schema, Error Status Condition Ø34Ø results.

- If AREA-ID is initialized with an area-name not included in the WITHIN clause in the Schema DDL, Error Status Condition Ø323 results.

- All data items, areas, records, and sets specified in a FIND statement must be defined in the invoked subschema; otherwise, Error Status Condition Ø346 results.

Error Status Codes For the FIND Statement

Condition	Content of ERROR-STAIUS
Area not open	0301
Database-key inconsistent with area-name	0302
Data items invalid or inconsistent	0304
Current record of set, area, record-name not known	0306
End of set or area or DUPs	0307
Implicitly referenced area not available	0318
Record not currently a member of named set	0322
Illegal area-name	0323
No record satisfies the FIND specified	0326
Attempted to find owner of a singular set	0333
Concurrent Access Error	0335
Record type not a member of named set	0340
Record type not owner of named set	0341
Record type not included in named area	0342
Arguments of Location Mode Clause not included in subschema	0343
Arguments of Set Occurrence Selection Clause not included in subschema	0344
Location mode of record not specified as CALC	0345
Specified data item, record, set, or area not in subschema	0346
Fatal - privacy breach attempted	-0304(0304F)

Examples and Discussion of FIND Statement

See the Examples and Discussion under Record Selection Expression.

****
* GET *
****

Function

Transfers the contents of the specified data items of the object record occurrence into the User Work Area for the invoked subschema.

General Format

GET [dbid-1 , dbid-2...].

Syntax Rules

1. Dbid-1 , dbid-2 ..., must be items defined in the subschema as being in the record type of the current record of the run-unit.

General Rules

1. The object of the GET statement is the current record of the run-unit.

2. The record-name of the object record serves as an implicit major qualifier for the data-base-identifiers. If they are not defined as part of the record type, Error Status Condition Ø5Ø4 results.

3. In cases where the format of a data item, as defined in the subschema invoked by the run-unit, differs from the definition of that data item in the schema:

   - The standard DBMS procedure will be in accordance with the rules specified in the PICTURE clause of the Schema DDL and the PICTURE specification of the Subschema DDL for COBOL.

4. Missing data items in the object record occurrence in the database will result in null-values being placed, for those data items, in the User Work Area. The value of a null item is as follows:

   - For numeric items, the item is set to zero.

   - For Alphanumeric DISPLAY items , the item is set to blanks.

5. A GET statement must be executed before any reference can be made to the data of the object record in the User Work Area.

6. If only "GET" is written, then all data items defined for the object record in the subschema invoked by the run-unit are moved to the User Work Area. If dbid's are specified, only the specified data items are moved to the User Work Area.

7. If the current record of the run-unit is not known, Error Status Condition Ø513 results.

8. If the run-unit has not satisfied the privacy locks on all data items and records needed to execute the GET, Error Status Condition Ø504F will result and the run-unit will be aborted by DBMS.

9. If the size of the dbid in the User Work Area is greater than the size of the item in the object record occurrence, the excess elements of the item in the User Work Area will be set to the appropriate null-value (see general rule 4).

1Ø. If the value of any data item in the database is such that it cannot be converted to the specified subschema format for that data item, Error Status Condition Ø519 results. (Conversion rules are included under the description of TYPE clause in the DDL for the SCHEMA and PICTURE and USAGE specification in the subschema DDL for COBOL.)

11. If any of the above Error Status Conditions is encountered, then the GET statement is not successfully executed. The record in the User Work Area is in an undefined state and the appropriate Error Status Condition code is made available in special register ERSTAT. Otherwise, ERSTAT is set to zero, indicating successful completion of the GET statement.

Error Status Codes For The GET Statement

Condition	Content of ERROR-STATUS
Data item invalid or inconsistent	0504
No current record of run-unit	0513
Conversion of value of data item not possible	0519
Fatal - privacy breach attempted	-0504(0504F)
Concurrent access error	0535
Specified dbid not included in subschema	0546

Examples and Discussion of the GET Statement

```
#    GET LNAME, FNAME, MNAME, DOB.
```

This command moves the contents of LNAME, FNAME, MNAME, DOB from the record to the User Work Area. It assumes that the current record of run-unit has these items defined within it and that the subschema has declared them. If necessary, conversion is performed on these items to change them from the schema type to the subschema type. Only the named items will be obtained from the record. The rest of the variables in the User Work Area are left unchanged,except ERSTAT,which is set to zero.

```
#    GET.
```

This command gets all the data items from the schema for the record type which is current of run-unit and places them in the User Work Area.

******
* IF *
******

Function

The IF statement causes a condition to be evaluated. The subsequent action of the run-unit depends on whether the value of the condition is true or false.

General Format

Format 1

```
IF set-name-1 SET [NOT] EMPTY { cobol-procedure-name-1 } [ ELSE cobol-procedure-name-2 ]
                              { NEXT                   }
```

Format 2

```
IF RECORD [NOT] { MEMBER } OF { set-name-2 } { cobol-procedure-name-3 }
                { OWNER  }    { ANY set    } { NEXT                   }

        [ELSE cobol-procedure-name-4].
```

Syntax Rules

none

General Rules

1. Set-name-1, set-name-2 must be defined in the invoked subschema; otherwise, Error Status Condition 1846 results.

2. Format 1 of the IF statement is designed to determine whether the object set occurrence has any members. The object set occurrence is determined by the current record of set-name-1. If the NOT phrase is omitted and the set occurrence does not have any member records, the condition is evaluated as true. If the NOT phrase is omitted and the set occurrence contains member records, the condition is evaluated as false. If the NOT phrase is stated, the condition is reversed.

3. Format 2 of the IF statement is designed to determine whether the current record of the run-unit currently participates as an owner or member, depending on the option specified, in set-name-2 or in any set. If the NOT phrase is omitted and the record is an owner or member as specified, the condition is evaluated as true. If the NOT phrase is omitted and the record is not an owner or member as specified, the condition is evaluated as false. If the NOT phrase is stated, the condition is reversed. If neither the OWNER nor MEMBER phrase is specified, the test is of the association as an owner or member of the object record with the specified set.

4. If the current record of the run-unit is not known, Error Status Condition 1813 results.

5. If the current record of the set-name specified is not known or the currency indicator is null, then the object set occurrence cannot be determined and Error Status Condition 1806 results.

6. If any of the above Error Status Conditions is encountered, then the IF statement is not successfully executed. The condition stated in the IF statement will be left unevaluated and the appropriate Error Status Condition code is made available in special register ERSTAT.

Error Status Codes for the IF Statement

Condition	Content of ERSTAT
Current of set not known	18Ø6
No current record of run-unit	1813
Specified set not included in subschema	1846

Examples and Discussion of the IF Statement

Example 1

```
#    IF DEPT-SET SET EMPTY NEXT ELSE LABEL-1.
     .
     .
     .
     .
     .
 LABEL-1.
     PERFORM NEXT-PARAGRAPH.
```

The IF command allows the application programmer to test for a number of conditions within the DBMS. The test in example one checks to see if there are any members of the "DEPT-SET". If the set is empty, control passes to the next statement; otherwise, control passes to the paragraph labeled LABEL-1.

Example 2

```
#    IF RECORD NOT MEMBER OF DEPT-SET LABEL-2.
```

In this example, the current record of run-unit is tested to see if it is a member of the DEPT-SET. If it is, control is passed to the next statement; otherwise, control is passed to the paragraph labeled LABEL-2.

****
* INSERT *
****

Function

Inserts and makes the object record a member of occurrences of the specified set-names, provided that it is defined as an optional automatic, optional manual, or mandatory manual member of those sets.

General Format

$$\underline{\text{INSERT}}\ \underline{\text{INTO}} \left\{ \begin{array}{l} \underline{\text{SETS}}\ \text{set-name-1}\ [,\text{set-name2}]\ldots \\ \underline{\text{ALL}}\ \text{SETS} \end{array} \right\} .$$

Syntax Rules

1. The sets, set-name-1, and set-name-2, must be defined in the invoked subschema.

2. The record that is current record of run-unit must be defined as an optional or manditory manual member of the specified sets. If the ALL option is used, the record must be of at least one of the sets in the invoked subschema.

General Rules

1. The object record occurrence of the INSERT statement is the current record of the run-unit.

2. If set-names are specified, then the object record must have been defined in the schema as an optional automatic, optional manual, or mandatory manual member of each set named. It will be inserted into the object set occurrence of each set-name specified in accordance with the set-ordering criteria defined in the schema. For each set named, the object set occurrence is determined by the current record of the named set.

- If the current record of any set-name specified is not known or its currency indicator is null, Error Status Condition Ø7Ø6 results.

3. The ALL SETS option inserts the object record into the appropriate occurrence of each set included in the invoked subschema, where it is defined as an optional automatic, optional manual, or mandatory manual member and in which it is not currently a member. The specific occurrence of each set will be determined by the current record of set-name for each of the set-names involved. The object-record will be inserted into each set occurrence in accordance with the set-ordering criteria specified in the schema.

   - If the current record of any set-name implicitly specified by ALL SETS is not known or its currency indicator is null, Error Status Condition Ø7Ø6 results.

4. As specified in the April 1971 DBTG Report, INSERT does not employ the set occurrence selection clauses defined in the schema which would otherwise have been aplicable for identifying the current occurrences of each set.

5. For each set-name into which the object record is inserted, it becomes the current record of set-name if such a currency update is not in a 'SUPPRESS' state (see the SUPPRESS command).

6. If the area in which the inserted record is stored is opened for concurrent update and there is a concurrent update conflict, Error Status Condition 735 results with CONTYP set to the concurrent update conflict type.

7. If the run-unit has not satisfied the privacy locks for all records, sets, areas, and data items needed to execute the INSERT statement, Error Status Condition Ø7Ø4F will result and the run-unit will be aborted by DBMS.

8. In addition to the conditions described, any of the following conditions will result in an error if they occur during the execution of an INSERT statement:

   - If all set-names specified are not included in the subschema, Error Status Condition Ø746 results.

   - If the current record of the run-unit is not known, Error Status Condition Ø713 results.

   - If the object record is not defined as an optional automatic, optional manual, or mandatory manual member of all of the specified set-names, Error Status Condition Ø714 results.

- If the object record, when inserted, would violate a DUPLICATES NOT ALLOWED clause for any record or set involved, Error Status Condition Ø7Ø5 results.

- If the current record of any set-name specified is not known or its currency indicator is null, Error Status Condition Ø7Ø6 results.

- If the object record is already a member of any occurrence of a set explicitly named in the INSERT statement, or if it is already a member of an occurrence of any set implicitly specified by the ALL SETS option, Error Status Condition Ø716 results. This is true whether the object record is a member in the object set occurrence or in any other occurrence of the same set.

- If the object record or any record occurrence affected by the INSERT statement is located in an area which is open for RETRIEVAL, Error Status Condition Ø7Ø9 results.

- If any record occurrence needed to execute the operation is located in areas that are not available, Error Status Condition Ø718 results. An area is not available if it is off-line or under the exclusive control of a concurrent run-unit.

9. When an error occurs, the database remains in the state existing prior to the attempted execution of the INSERT statement and the appropriate Error Status Condition code is made available in special register ERSTAT.

Error Status Codes for the INSERT Statement

Condition	Content of ERROR-STATUS
Violation of DUPLICATES NOT ALLOWED clause	0705
Current record of set-name not known	0706
Incorrect usage mode for area	0709
No current record of run-unit	0713
Object record not defined as an optional member or mandatory manual member of a named set	0714
Record already a member of named set	0716
Implicitly referenced area not available	0718
Concurrent update conflict	0735
Specified set-name not in subschema	0746
Fatal-Privacy Breach Attempted	-0704(0704F)

Examples and Discussion for the INSERT Statement

```
#    INSERT INTO ALL SETS.
```

The current record of run-unit is inserted into all sets of which it is not already a member. In addition, if the sets have search keys defined, the search key is inserted into its list. The insertion into the set is based on the ordering criteria given in the schema.

****
* INVOKE *
****

Function

INVOKE is used to indicate that the COBOL program is ready to use the DBMS services and to establish the program as a DBMS user.

It is also used to invoke the use of the subschema specified in the subschema declaration.

General Format

INVOKE DBMS.

Syntax Rules

None

General Rules

1. The INVOKE statement must be executed before any other DML statement. Before it is executed, all other DML statements executed by the COBOL program will be ignored by DBMS.

2. The INVOKE statement causes a sequence to establish the COBOL program as a run-unit under Prime DBMS.

3. If the schema or the subschema which the run-unit wishes to use is not available to the DBMS, a fatal error occurs and the run-unit will be aborted.

Error Status Codes for the INVOKE Statement

Condition	Content of ERSTAT
Invoke has already been executed	1403
Fatal - invoked subschema has been deleted from DBMS	-1401(1401F)
Fatal - invoked schema has been deleted from DBMS	-1401(1402F)

Examples and Discussion

None.

****
* MODIFY *
****

Function

The MODIFY statement replaces the values of all, or of specific data items of the object record occurrence in the database, with values from the User Work Area.

General Format

MODIFY [dbid-1 , dbid-2...].

Syntax Rules

1. dbid-1, dbid-2..., must be defined as items of the record which is a current record of run-unit in the invoked subschema.

General Rules

1. The object record occurrence of the MODIFY is the current record of the run-unit. If the current record of the run-unit is not known, the MODIFY statement is not executed and Error Status Condition 0813 results.

2. The record-name of the object record serves as an implicit major qualifier for the data-base-identifiers. They must be defined as part of the record type, otherwise Error Status Condition 0804 results.

3. If data-base-identifiers are not specified, all data items in the object record which are specified in the subschema named in the INVOKE clause of the program are modified with (i.e., replaced by) values from the User Work Area. Any data item in the object record occurrence in the database which is not specified in the invoked subschema remains unchanged.

4. If data-base-identifiers are specified, only the data items specified are modified with values from the User Work Area. All other data items in the object record occurrence in the database remain unchanged.

5. In cases where the definition of a data item in the subschema invoked by the run-unit differs from the definition of that data item in the schema. Conversions performed by DBMS will be in accordance with the rules specified in the TYPE clause of the Schema DDL and the TYPE specification of the subschema DDL for COBOL.

6. If any of the modified data items is defined as a sort-control item in the object record for any set occurrence in which the object is currently a member, then its modification causes the intra-set occurrence position of the object record to be examined. If necessary, the object record is removed and re-inserted in the set occurrence to maintain the set order specified in the schema. The current occurrence of the set-name involved remains as the current occurrence.

   If the current of run-unit is the current of the set-name involved, it remains as the current of set-name. If it is not, it becomes the current of that set-name if such a currency update is not in a 'SUPPRESS' state (see the SUPPRESS command). The database-key of the object record remains unchanged.

7. If any of the modified data items are defined with a SEARCH KEY clause in any set in which the object record is currently a member, then the execution of the MODIFY statement causes DBMS to adjust the indices it maintains for those data items. The database-key of the object record remains unchanged.

8. If any of the modified data items is defined as a CALC key in the LOCATION MODE clause for the object record, then the execution of the MODIFY statement causes the DBMS to make the necessary adjustments which will enable the record to be found on the basis of the new values for the CALC keys. The data item (if any) specified as AREA-ID must also be initialized with an area-name specified or implied for the record type; otherwise Error Status Condition Ø823 results.

9. If the insertion of the object record under any of the conditions described in general rules 6, 7, or 8 would violate a DUPLICATES NOT ALLOWED clause (defined for any of the sets or records involved), then the MODIFY statement is not executed and Error Status Condition Ø8Ø5 results.

1Ø. All data items involved must be initialized in the User Work Area with the required values prior to execution of the MODIFY statement.

11. If the run-unit has not satisfied the privacy locks for all records, sets, areas, and data items needed to execute the MODIFY, Error Status Condition Ø8Ø4F will result and the run-unit will be aborted by DBMS.

12. If the object record or any record occurrence affected by the execution of the MODIFY statement is located in an area which is open for RETRIEVAL, Error Status Condition Ø8Ø9 results.

13. If the area in which the modified record is opened for concurrent update and there is a concurrent update conflict Error Status Condition Ø835, results and CONTYP is set to the concurrent update conflict type.

14. If any record occurrence needed to execute the operation is located in areas that are not available, Error Status Condition Ø818 results. An area is not available if it is off-line or under the exclusive control of a concurrent run-unit.

15. If the value of a data item in the User Work Area is such that it cannot be converted to the format specified in the schema for that data item, Error Status Condition Ø819 results.

16. Under all Error Status Conditions described in the above general rules, the database remains in the state existing prior to the attempted execution of the MODIFY statement, and the appropriate Error Status Condition code is made available in special register ERSTAT. Otherwise, ERSTAT is set to zero, indicating successful execution of the MODIFY statement.

Error Status Codes for the MODIFY Statement

Condition	Content of ERROR-STATUS
Database-key inconsistent with area-name	0802
Data items invalid or inconsistent	0804
Violation of DUPLICATES NOT ALLOWED clause	0805
Incorrect usage mode for area	0809
No current record of run-unit	0813
Implicitly referenced area not available	0818
Conversion of value of data item not possible	0819
Illegal area-name	0823
Concurrent update conflict	0835
Arguments of Location Mode clause not included in subschema	0843
Fatal - Privacy Breach Attempted	-0804(0804F)
Arguments of Set Occurrence Selection clause not included in subschema	0844
Specified set not included in subschema	0846

Examples and Discussion of the MODIFY Statement

Example 1

```
#    MODIFY LNAME, FNAME.
```

The MODIFY command allows the application programmer to change the contents of a given record. The MODIFY command always operates on the current record of run-unit. In the example above, the fields LNAME, FNAME in the record will be replaced by the contents of the fields in the User Work Area. The items named must be defined in the subschema and be contained in the record that is current record of run-unit.

In addition to changing the contents of the fields, if the items make up any part of a sort, the list is reorded to reflect the new ordering with the new value. If the item is a CALC key, the record is recalced to reflect the value of the new key.

Example 2

```
#    MODIFY.
```

This form of the MODIFY command replaces the entire contents of the record, which is current of run-unit, with the contents of the User Work Area. It does all conversions called for between schema types and subschema types. It also re-orders any set list which contains items that are sort or search keys, and if the record has a location mode CALC, the record is recalced to reflect the new value of the CALC keys.

****
* MOVE *
****

Function

The MOVE statement saves the contents of the specified currency status indicators and provides a means for deriving the area-name or record-name which corresponds to a database-key.

General Format

Format 1

```
MOVE CURRENCY STATUS FOR { RUN-UNIT
                         { RECORD record-name
                         { AREA area-name
                         { SET set-name } TO identifier-1 .
```

Format 2

```
MOVE { RECORD-NAME } FOR { RUN-UNIT
     { AREA-NAME   }     { RECORD record-name
                         { AREA area-name
                         { SET set-name
                         { identifier-2 } TO identifier-3 .
```

Syntax Rules

1. The specified record-name, area-name or sets name must be included in the invoked subschema.

2. Identifier-1 and identifier-2 must refer to data items that are used as Database keys. Identifier-3 must refer to a character data type.

General Rules

1. The specified record-name, area-name, or set-name must be included in the invoked sub-schema; otherwise, Error Status Condition 1946 results.

2. Identifier-1 and identifier-2 are assumed to be level 77 fields with a length of 6 characters (i.e., COBOL variables meant to contain database-keys). For more detail about the representation

of a database-key, refer to the TYPE clause in the Schema DDL and the TYPE specification in the Subschema DDL for COBOL (See Section 2).

3. Identifier-3 is assumed to be a level 77 variable which can contain a 30 character string.

4. In Format 1, if the RUN-UNIT phrase is specified, the database-key for the current record of run-unit is placed in identifier-1. The current record of the run-unit is not altered. If a record-name, area-name, or set-name is specified, the database-key for the current record of record-name, area-name, or set-name is placed in identifier-1. The current record of record-name, area-name, or set-name is not altered. If the current record is not known or the currency indicator is null, Error Status Condition 1913 results.

5. Use of Format 2 causes DBMS to return, in identifier-3, the name of the area or record which corresponds to the database-key in identifier-2 or to the currency indicator specified. If a currency indicator is specified and the current record is not known or the currency indicator is null, Error Status Condition 1913 results.

6. When an error occurs, the database and all of the run-unit's COBOL variables remain in the state existing prior to the attempted execution of the MOVE statement.

Error Status Codes for the MOVE Statement

Condition	Content of ERSTAT
No current record of run-unit	1913
Specified record, area, or set not included in subschema	1946

Examples and Discussion of the MOVE Statement

Example 1

```
#    MOVE CURRENCY STATUS FOR RECORD EMPLOYEE TO ID1.
```

This statement takes the database-key which is the currency status for the record EMPLOYEE and moves it to the COBOL identifier ID1. The run-unit can later refer to the record by a

```
#    FIND ID1.
```

To establish the record as current of run-unit.

Example 2

```
#    MOVE AREA-NAME FOR ID1 TO ANAME.
```

This statement gets the 3Ø-character Area-name of the DBK defined in ID1 and moves it to the 15-word COBOL array ANAME.

****
* ON ERROR CLAUSE *
****

Function

The ON ERROR clause specifies where program control is to be transferred if the DBMS encounters an error during the execution of a DML command.

General Format

Format 1

```
;ON ALL ERRORS GO TO cobol-procedure-name .
```

Format 2

```
;ON ERROR integer-1 [,integer-2]... GO TO cobol-procedure-name ... .

    [ON ERROR integer-3[,integer-4] ... GO TO cobol-procedure-name]... .

    [ON OTHER ERRORS GO TO cobol-procedure-name].
```

Syntax Rules

1. integer-1, integer-2,... are COMPUTATIONAL values of the form:

   MMEE

   where MM is the major code of the command

   EE is the error code

   The major code is an optional specification

2. No check is ever made to verify that the values specified by integer-1, integer-2,... can actually occur. If an invalid code is used, no branch will ever be taken at run time.

3. Label must be a valid COBOL procedure-name known to this program.

General Rules

1. The ON ERROR clause may be appended to any DML command.

2. If an error is detected, the DBMS returns to the statement specified by the appropriate ON ERROR clause (if any).

General Rules

1. The ON ERROR clause may be appended to any DML command.

2. If an error is detected, the DBMS returns to the statement specified by the appropriate ON ERROR clause (if any).

3. The ON ERROR clause refers only to the command of which it is a part.

4. If an error is detected and there is no error clause, control is returned to the run-unit at the statement following the DML command. The run-unit may test the register ERSTAT to see if an error occurred. If ERSTAT is 0, then no error has occurred; otherwise, ERSTAT will be of the form MMEE

   where MM is the major code

   EE is the error code

5. If an error has occurred, no further DML statements may be processed until a CLEAR ERROR command has been executed.

6. If another DML statement is executed before a CLEAR ERROR command is executed, or if a fatal error has occurred, the run-unit is immediately aborted and the special registers are dumped to the terminal.

7. Whenever the execution of a DML statement results in an Error Status Condition, the following information is available to the run-unit:

   - The Error Status Condition code is available in the special COBOL COMPUTATIONAL variable ERSTAT.
   - The name of the area in which the error occurred is available in the 30 character field ERAREA.
   - The name of the set, if appropriate, in which the error occurred is available in the 30 character field array ERSET.
   - The name of the record, if appropriate, for which the error occurred is available in the 30 character field ERREC.
   - If ERSTAT contains an error code of nn35 which indicates a concurrent update conflict, then the register CONTYP contains a code indicating the type of concurrent update conflict.

- The name of the item, if appropriate, for which the error occured is available in the 30 character field ERITEM.
- ERCASE contains the conditional value of an error return for a GO TO DEPENDING ON statement.
- The number of the current transaction can always be found in TRNO.
- The type of the current transaction can be found in TRTYPE where 1=update transaction and Ø=retrieval transaction.

8. A summary of the major codes, error codes and concurrent update conflict codes may be found in Section 4.

Examples and Discussion for the ON ERROR Clause

Example 1

```
#     FIND NEXT RECORD EMPLOYEE OF SET DEPT-SET
          ON ERROR 307 GO TO LABEL-1.

#    GET.
       .
       .
       .
 LABEL-1.
     DISPLAY 'END OF SET ENCOUNTERED'.
```

This example shows a find statement with an error clause. If an end of set is encountered, control will pass to LABEL-1; otherwise control will pass to the next statement. If an error other than 307 occurred, no other CDML statement may be executed until a clear error command has been executed.

Example 2

```
#     FIND RECORD DEPT.
    IF ERSTAT NOT = Ø GO TO LABEL-1.
       .
       .
       .
 LABEL-1.
     DISPLAY 'DBMS ERROR' ERSTAT.
#    CLEAR ERROR.
```

Example 2 demonstrates an alternate form of DBMS error detection. If the FIND in the example succeeds and there are no errors, ERSTAT will be Ø and control will proceed to the next statement. Otherwise, ERSTAT will be nonzero and control will be passed to TABLE-1 where the error

is written to the user terminal and the error status is reset by the CLEAR ERROR command.

****
* OPEN *
****

Function

The OPEN statement specifies an area's usage-mode and prevents the run-unit's access to the area until such usage can be permitted.

General Format

Format 1

```
OPEN ALL AREAS [ USAGE-MODE IS [ EXCLUSIVE ] { RETRIEVAL } ] .
               [ PROTECTED ]  { UPDATE    }
```

Format 2

```
OPEN AREA[S] area-name-1 [,area-name-2]...
     [ USAGE MODE IS [ EXCLUSIVE ] { RETRIEVAL } ] .
                     [ PROTECTED ] { UPDATE    }
```

Syntax Rules

1. Areas area-name-1, area-name-2, must be names which are included in the invoked subschema.

General Rules

1. Area-name-1, area-name-2, must be the names of areas included in the subschema invoked; otherwise, Error Status Condition Ø946 results.

2. The OPEN ALL format refers to all the areas included in the subschema invoked by the run-unit.

3. Because of the file-level deadlock prevention plan of DBMS, an OPEN can only be executed if the run-unit has no areas already open, otherwise Error Status Ø929 results. Thus you must open any areas together that will be needed simultaneously; you are requested, additionally, to open no more areas than you will need at one time, and to close all areas during any relatively long interval when you are not using DBMS services. The CLOSE statement, unlike the OPEN, may close a subset of the areas open.

4. Use of the USAGE-MODE IS RETRIEVAL phrase (without the EXCLUSIVE or PROTECTED phrases) allows concurrent run-units to open the same area with any usage-mode other than one which is exclusive.

5. Use of the USAGE-MODE IS UPDATE phrase (without the EXCLUSIVE or PROTECTED phrases) allows concurrent run-units to open the same area with any usage-mode other than one which is exclusive or protected.

6. If the USAGE-MODE phrase is not used, then retrieval without the exclusive or protected option, called "concurrent retrieval", is assumed.

7. Use of the EXCLUSIVE phrase prevents concurrent run-units from interacting with the same area in any usage-mode.

8. Use of the PROTECTED phrase prevents concurrent update and allows concurrent retrieval within the same area if and only if before-imaging is turned off and concurrent update is not allowed by the Data Administrator. Otherwise, protected update and protected retrieval are the same as simple update and simple retrieval.

9. All specified usage-modes remain in effect until the run-unit issues a CLOSE statement for the specified areas, or until the run-unit terminates.

1Ø. To execute a FIND, STORE, MODIFY, INSERT, REMOVE or DELETE statement successfully, the run-unit must have previously opened (as relevant to the statement being executed):

   - The area that contains the object record of a FIND statement,
   - The area into which a record is to be stored.

- All areas containing any record occurrence which would be deleted or removed as a result of a DELETE statement. Otherwise, Error Status Condition nnØ1 results, where nn indicates the particular DML statement being attempted.

11. In addition to the areas containing the object records of the statements cited above, there are additional, implicit areas which can be impacted by DML statements. The impact can be of two forms:

    - DBMS requires information contained within the implicit area, in which case the area must be 'available'. For an area to be available, it must not be under exclusive control of a concurrent run-unit. It must be OPEN for the requesting run-unit. If an implicit area is not available, Error Status Condition nn18 results (where nn indicates the particular DML statement attempted.)

    - The DBMS must alter the information contained in record occurrences within the implicit area, in which case the area must not only be available but it must permit the necessary alterations. Such implicit areas are deemed to be 'affected', and must be OPEN. If an implicit area which is affected is not open, Error Status Condition nn21 results (where nn indicates the particular DML statement attempted).

12. Record occurrences which are in the search path of the sought record of a FIND statement, or in the search path of the sought record of an implicit find which occurs during the execution of a STORE statement, must be in areas which are open.

13. To execute an INSERT, REMOVE, STORE, DELETE, or MODIFY statement successfully, both the explicit and the affected implicit areas involved must be open with a usage-mode of update. If any of the areas involved are open for retrieval, Error Status Condition nnØ9 will result, (where nn indicates the particular DML statement attempted.)

14. Any attempt to execute an OPEN statement which would result in usage-mode conflict for an area will result in Error Status Condition Ø931. The user can then choose to program a wait sequence or go on to do something else.

    - The following table reflects usage-mode conflicts with an 'N', and permitted concurrency with a 'Y'.

    - Usage-mode conflict occurs when any of the usage-mode combinations marked with an 'N' applies to an area as a result of a concurrent run-unit attempting to execute an open against it.

    - In the shaded areas (see Figure 3-1) the usage-mode conflict occurs only if before-imaging is turned off.

USAGE-MODE	OPEN-NO USAGE-MODE	RETRIEVAL	UPDATE	PROTECTED RETRIEVAL	PROTECTED UPDATE	EXCLUSIVE RETRIEVAL	EXCLUSIVE UPDATE
OPEN-NO USAGE-MODE	Y	Y	Y	Y	Y	N	N
RETRIEVAL	Y	Y	Y	Y	Y	N	N
UPDATE	Y	Y				N	N
PROTECTED RETRIEVAL	Y	Y		Y		N	N
PROTECTED UPDATE	Y	Y				N	N
EXCLUSIVE RETRIEVAL	N	N	N	N	N	N	N
EXCLUSIVE UPDATE	N	N	N	N	N	N	N

FIGURE 3-1. USAGE-MODE RULES

15. An attempt to execute an OPEN statement on an area for which the run-unit has not satisfied the privacy locks will result in Error Status Condition 0904F and the user will be aborted.

16. If none of the Error conditions described above occurs, ERSTAT is zero, indicating successful execution of the OPEN statement.

17. If the total number of areas open by all concurrent run-units exceeds an implementation limit, Error Status 0905F results and the user will be aborted.

Error Status Codes for the OPEN Statement

Condition	Content of ERSTAT
Area not physically available	Ø928
Violation of deadlock protection rule	Ø929
Open blocked	Ø931
Specified area-name not in subschema	Ø946
Fatal - privacy breach attempted	-Ø9Ø4(Ø9Ø4F)
Fatal - attempted to open too many areas simultaneously	-Ø9Ø5(Ø9Ø5F)

Examples and Discussion for the OPEN Statement

Example 1

```
#    OPEN ALL AREAS.
```

This command will make available all areas which have been defined in the subschema, for a usage- mode of simple retrieval.

Example 2

```
#    OPEN AREAS AREA-1, AREA-2, USAGE-MODE IS EXCLUSIVE UPDATE AREAS
     AREA-3, AREA-4 USAGE MODE IS RETRIEVAL.
```

This command will make available AREA-1 and AREA-2 for exclusive update and AREA-3 and AREA-4 for concurrent retrieval. This is the only way different areas are allowed to be open for different usage-modes. The execution of two open commands would violate the file-level dead lock prevention scheme in the DBMS.

Example 3

```
#    OPEN ALL AREAS USAGE-MODE IS PROTECTED RETRIEVAL.
```

This form of the open command would open all areas defined in the subschema. The actual usage mode depends on several factors. If concurrent update mode and before-imaging are turned off via the Data Administrator module, then the usage mode would be protected retrieval.

If concurrent update were allowed and before-imaging was on, then it is possible that there will be concurrent updaters but the application program would be guaranteed a consistent view of the database for the duration of each transaction.

****
* PRIVACY KEY *
****

Function

The PRIVACY KEY statement is executable, not declarative and establishes the program and run-unit authority to execute classified DML imperative-statements in accordance with the locks declared in the schema.

General Format

Format 1

```
PRIVACY KEY [FOR [ [EXCLUSIVE] RETRIEVAL ]
                [ [PROTECTED]           ]
                [ [EXCLUSIVE] UPDATE    ]
                [ [PROTECTED]           ] ]OF

{ AREAS area-name-1 [,area-name-2]... } IS { literal-1    }.
{ ALL AREAS                           }    { identifier-1 }
```

Format 2

```
PRIVACY KEY [FOR { REST                   }
                 { STORE                  }
                 { GET                    }
                 { MODIFY                 }
                 { INSERT                 }
                 { REMOVE                 }
                 { DELETE MANDATORY       }
                 { DELETE SELECTIVE       }
                 { DELETE ALL             }
                 { FIND                   } ] OF

{ RECORDS record-name-1 [,record-name-2]... } IS { literal-2    }.
{ ALL RECORDS                                }    { identifier-2 }
```

Format 3

```
PRIVACY KEY [FOR { REST   }
                 { STORE  } ]OF DATA-ITEMS dbid-1 [,dbid-2]...
                 { GET    }
                 { MODIFY }

                                IS { literal-3    }.
                                   { identifier-3 }
```

Format 4

```
PRIVACY KEY [FOR { REST   } ] OF { SETS set-name-1 [,set-name-2]... }
                 { INSERT }      { ALL SETS                         }
                 { REMOVE }
                 { FIND   }

                IS { literal-4    } .
                   { identifier-4 }
```

Syntax Rules

1. Multiple PRIVACY clauses that differ only in the specification of the key are not permitted.

2. All identifiers must be defined in the COBOL source program as level 77 identifiers.

General Rules

1. Identifier-1, 2, 3, and 4 are intended to contain PRIVACY KEYS.

2. If the optional FOR phrase is omitted, the PRIVACY KEY clause applies to all functions that could have been specified.

3. Any database-identifiers, records, sets, or areas named which are not included in the invoked subschema will be ignored.

4. If a PRIVACY LOCK has been declared in the schema, a PRIVACY command must be executed at run time to allow usage of the function even if no lock value has been assigned.

5. If the specified area-name, record-name, set-name or dbid-name are not in the invoked subschema, Error Status 1346 results.

Examples and Discussion for the PRIVACY KEY

```
       WORKING-STORAGE SECTION.
      #    SUBSCHEMA MY-SUB OF SCHEMA THE-SCHEMA.
       01  LOCK-VALUE    USAGE DISPLAY PIC X(30).

       .
       .
       .
       .


       PROCEDURE DIVISION.
       .
       .
       .
           DISPLAY 'ENTER KEY FOR OPEN WITH USAGE-MODE OF UPDATE'.
           ACCEPT LOCK-VALUE.
      #    INVOKE DBMS.
       .
       .

      #    PRIVACY KEY FOR UPDATE OF ALL AREAS IS LOCK-VALUE.
      #    OPEN ALL AREAS USAGE-MODE IS UPDATE.
```

This example illustrates the use of a PRIVACY command in a run-unit. The sample program reads a 30-character string from the user terminal; it then invokes the DBMS and passes the key to unlock the open for update. If an improper key is passed, the open command will fail. If the proper key is passed or the Data Administrator has not yet defined the key value, the open will succeed.

The PRIVACY KEY command can be executed any time before the entity with a lock defined for a particular operation is accessed in that type of operation.

****
* RECORD SELECTION EXPRESSIONS (rse) *
****

Record selection expressions are used to specify the criteria by which DBMS is to select a record in the database. The selected record becomes the current record at the run-unit, upon which subsequent statements may operate when accessing the database.

There are six formats of the record selection expression. These formats permit both relative and absolute selection criteria in which records may be selected according to their contents, their database key value, or by currency indicators. If a currency indicator is used, either a current record or a record related to a current record is selected.

The record selection expression may only occur on a FIND or a FETCH statement.

General Format

Format 1

```
USING identifier-1 .
```

Format 2

```
[ { OWNER  } IN set-name OF ] CURRENT OF { RECORD record-name-2 }
  { MEMBER }                             { SET set-name-4       }
                                         { AREA area-name-1     }
                                         { RUN-UNIT             } .
```

Format 3

```
{ NEXT       }
{ PRIOR      }
{ FIRST      } RECORD [record name-3] OF { SET set-name-5 }
{ LAST       }                           { AREA area-name-2 } .
{ integer-1  }
{ identifier-2 }
```

Format 4

```
[NEXT DUPLICATE WITHIN] RECORD record-name-4 .
```

Format 5

```
record-name-5 VIA [CURRENT OF] SET set-name-7 [USING dbid-3 [,dbid4]...] .
```

Format 6

```
NEXT DUPLICATE WITHIN SET set-name-8 USING dbid-5 [,dbid-6]... .
```

Syntax Rules

Format 1:

1. Identifier 1 must be a six byte level 77 identifier containing a database key.

Format 2:

1. Record-name-2 if specified must be defined in the subschema.

Format 3:

1. Integer-1 may be signed.
2. The data item referenced by identifier-2 must be USAGE COMPUTATIONAL and may be signed or unsigned.
3. If both record-name-3 and set-name-5 are used, then the record must be a member of the named set.

Format 4:

1. The record named by record-name-4 must have location mode CALC.

Format 5:

1. dbid-3, dbid-4..., must be items defined within record-name-5.

Format 6:

1. dbid-5, dbid-6..., must be items defined within the record which is the current record of the set type specified in set-name-8.

General Rules

All formats

1. Evaluation of a record selection expression results in the identification by the DBMS of a specific record in the database.
2. Records identified by a record selection expression must be stored in areas that are in an opened mode.
3. All records that participate in the DBMS search for the identified record must be stored in areas that are opened.

4. Error Statuses defined in the following rules of the form nnXX may have the values Ø3XX if the error occurred on a FIND, or 22XX if the error occurred on a FETCH.

Format 1:

1. The record identified is the record whose database-key is equal to the value of the data item referenced by identifier-1.

Format 2:

1. This clause without the OWNER or MEMBER phrase selects the record occurrence that is the current record of the specified record-name, set-name, or area-name, or it selects the current record of run-unit.

2. If the current record of the type specified is not known or its currency indicator is null, Error Status Condition nnØ6 results.

3. Use of the CURRENT OF RUN-UNIT form of the FIND permits revision of currency status indicators which were previously suppressed.

4. Set-name-3 and set-name-4 may be the same set-name or different set-names.

5. If the OWNER phrase is used, the occurrence of set-name-3 whose owner is to be selected is identified by the specified current record. Set-name-3 must not be a singular set; otherwise, Error Status Condition nn33 results.

6. If the specified current record is not of a record type defined as a member of the set in the schema, Error Status Condition nn4Ø results.

7. If the specified current record does not currently participate as a member of any occurrence of set-name-3, and if it is not known as the current record of set-name-3 by the run-unit, Error Status Condition nn22 results.

8. If the MEMBER phrase is used, the record selected is the first member (in terms of logical order of the set) in the set occurrence of set-name-3 owned by the specified current record.

9. If the specified current record is not of the record type defined to be the owner record type of set-name-3 in the schema, Error Status Condition nn41 results.

1Ø. If the set occurrence of set-name-3 owned by the specified current record is null, Error Status Condition nn26 results.

Format 3:

1. In Format 3, if the record-name-3 phrase is used, only occurrences of record-name-3 will be considered in evaluating the record selection expression. If area-name-2 is stated, then the WITHIN clause in the schema for record-name-3 must include area-name-2; otherwise, Error Status Condition nn42 results. If set-name-5 is stated, record-name-3 must be defined as a member of set-name-5; otherwise, Error Status Condition nn4Ø results.

2. If a set-name is specified, the set occurrence from which the object record is to be selected is identified by the current record of the specified set. If the current record of that set is not known or its currency indicator is null, Error Status Condition nnØ6 results.

3. If an area-name is specified, the object record is selected from the named area. If the named area is not open, Error Status Condition nnØ1 results.

4. If the NEXT or PRIOR phrase is used and an area-name is specified, and the current record of the named area is not known or its currency indicator is null, Error Status Condition nnØ6 results.

5. NEXT RECORD OF area-name AREA means the record physically closest in the "next" direction to the current record of the named area. If there is no such record in the area named, Error Status Condition nnØ7 results.

6. PRIOR RECORD OF area-name AREA means the record physically closest in the "prior" direction to the current record of the named area. If there is no such record in the area named, Error Status Condition nnØ7 results.

7. NEXT RECORD OF set-name SET means the subsequent record relative to the current record of the named set in the logical order of the set regardless of the database-key sequence. If the set is empty, that is, no member record occurrences participate in the set, Error Status Condition nn26 results. If the current record is the last record in the set, Error Status Condition nnØ7 results.

8. PRIOR RECORD OF set-name SET means the previous record relative to the current record of the named set in the logical order of the set regardless of database-key sequence. If the set is empty, that is, no member record occurrences participate in the set, Error Status Condition nn26 results. If the current record is the first record in the set, Error Status Condition nnØ7 results.

9. FIRST RECORD OF area-name AREA is the record occurrence physically first in the named area. If there are no records in the named area, Error Status Condition nn26 results.

1Ø. LAST RECORD OF area-name AREA is the record occurrence physically last in the named area. If there are no records in the named area, Error Status Condition nn26 results.

11. FIRST RECORD OF set-name SET is the first member occurrence in terms of the logical order of the set. The record selected is the same as would be selected if the current record of the set was the owner record and the NEXT RECORD OF set-name SET was used. If the set occurrence is empty, that is, no member record occurrences participate in the set, Error Status Condition nn26 results.

12. LAST RECORD OF set-name SET is the last member record occurrence in terms of the logical order of the set. The record selected is the same as would be selected if the current record of the set was the owner record and the PRIOR RECORD OF set-name SET was used. If the set occurrence is empty, that is, no member record occurrences participate in the set, Error Status Condition nn26 results.

13. Identifier-2 must be USAGE COMPUTATIONAL and be initialized with an integer prior to execution of the FIND statement. Identifier-2 and integer-1 represent the ordinal count of the object record occurrence relative to the beginning, if positive, or ending, if negative of a set occurrence or area. A negative value selects in the prior direction and a positive value in the next direction of the set occurrence or area.

14. If there are no records in the set occurrence or area named, Error Status Condition nn26 results. If the value of integer-1 or the contents of identifier-2 are greater than the number of record occurrences in the set occurrence or area specified, then Error Status Condition nnØ7 results.

Format 4:

1. In Format 4, the LOCATION MODE IS CALC clause must have been used in the description of record-name-4; otherwise, Error Status Condition nn45 results. All search arguments specified in the CALC clause must be included in the invoked subschema; otherwise, Error Status Condition nn43 results.

2. Prior to the execution of the FIND statement, the data items specified in the LOCATION MODE clause must be initialized and if the related WITHIN clause includes more than one area, the data item declared as AREA-ID must also be initialized with the area-name to which the resulting database-key applies. If an area-name is specified which is not included in the WITHIN clause, Error Status Condition nn23 occurs. If no record with the specified CALC key values is found, Error Status Condition nn26 results.

3. When the NEXT DUPLICATE phrase is not used, the first record found by DBMS which satisfies the argument values for the CALC keys in User Work Area is the one selected. When the NEXT DUPLICATE phrase is used and the following conditions exist, the record occurrence selected will be the next record found by DBMS with the same value for its CALC key as the current record of the run-unit. If no such duplicate record occurrence is found, then Error Status Condition nn26 results.

   - The CALC key in UWA is equal to the value of the CALC key in the current record of run-unit.

   - The implied area-name or the area-name explicitly specified in AREA-ID is the area in which the current record of run-unit exists.

4. When the NEXT DUPLICATE phrase is used and the above conditions do not exist, the effect is the same as if the NEXT DUPLICATE phrase had not been stated.

Format 5:

1. Record-name-5 must be defined in the schema as a member of set-name-7; otherwise, Error Status Condition nn40 results.

2. Format 5 of the rse, when the CURRENT phrase is used, causes record selection based on the current set occurrence of set-name-7. When the CURRENT phrase is not used it causes record selection based on the SET OCCURRENCE SELECTION clause defined for the named record and set. All search arguments specified in that clause must be included in the invoked subschema; otherwise, Error Status Condition nn44 results. The search arguments must be initialized prior to the execution of the FIND statement. In addition, if the process involves selection of any record on the basis of a LOCATION MODE IS CALC clause, any AREA-ID in the WITHIN clause for that record must be initialized. If AREA-ID is not initialized with an area-name specified or implied for the record type, Error Status Condition nn02 results. If the database-key developed from the CALC keys is inconsistent with the area-name specified, Error Status Condition nn02 results.

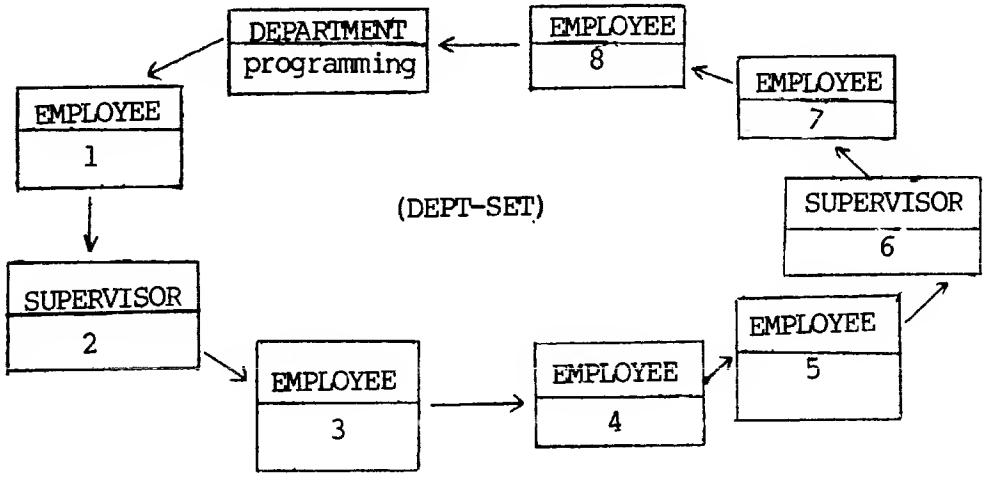
3. The record selected if the USING phrase is not specified is the first record occurrence of the record type specified by record-name-5, which is in the selected occurrence of the named set. If the USING phrase is employed, data-base-identifier-3 and data-base-identifier-4 are implicitly qualified by record-name-5 and must be the names of fixed length data items included in record-name-5; otherwise, Error Status Condition nn04 results. DBMS will select a record having the same values for data-base-identifier-3 and data-base-identifier-4 as are in the User Work Area. If more than one record occurrence meets the criteria specified, the first such record occurrence will be selected in terms of scanning the set in the next direction. If no record is found which meets these criteria, Error Status Condition nn26 results. When the CURRENT phrase is used, if the current record of set-name-7 is not known or its currency indicator is null, Error Status Condition nn06 results.

Format 6:

1. Format 6 of the rse causes a search of the members of the current set occurrence of set-name-8 for a record which is of the same type as the current record of set-name-8, and, which has the same values for data-base-identifier-5, data-base-identifier-6. All values in the User Work Area are ignored. The search is in the next direction and starts from the current record of set-name-8. It continues until either a duplicate is found or the end of set is reached. If no duplicate is found, Error Status Condition nn07 results. If the current record of set-name-8 is not known or its currency indicator is null, Error Status Condition nn06 results. If the current record of set-name-8 has been deleted by this or another run-unit, since it became current in this run-unit, Error Status Condition nn17 results. If data-base-identifier-5, data-base-identifier-6 are not defined in the subschema invoked as part of the current record of set-name-8, Error Status Condition nn04 results.

Examples and Discussion of the RECORD SELECTION EXPRESSIONS

For the following examples assuming there is a database with these record types: A DEPARTMENT (DEPT) record, an EMPLOYEE record and a SUPERVISOR record. The DEPT record has a location mode of CALC on the DEPTNM field and is the owner of the DEPARTMENT-SET (DEPT-SET) of which the EMPLOYEE and the SUPERVISOR records are a member. The set is sorted on an EMPLOYEE NUMBER field which is part of both the EMPLOYEE record and the SUPERVISOR record. The database has been pre-stored with the following set occurrence.



Example 1

```
#    FIND USING DBK1.
```

The COBOL identifier DBK1 has been declared to be PICTURE X(6) which has been initialized to a valid database-key. The result of this command is that the record specified by this DBK becomes current at the run-unit area in which it is stored, of record and of all sets in which it participates as an owner or member.

Example 2

```
#    FIND OWNER IN DEPT-SET OF CURRENT OF RECORD EMPLOYEE.
```

Assuming that the current of RECORD Employee is EMPLOYEE 4, the owner record DEPARTMENT which is programming will be selected and become current at run-unit, current at record type DEPARTMENT, current at set DEPARTMENT-SET and current at the area in which the DEPT record resides.

Example 3

```
#    FIND -2 RECORD SUPERVISOR OF SET DEPT-SET.
```

The format of the rse will traverse the DEPT-SET in the prior direction and produce the supervisor record with the number 2 as the current of run-unit at all. It should be noted that the employee record in the set did not affect the count. Had the "RECORD SUPERVISOR" clause been dropped from the FIND, the Employee Record 8 would be the record selected.

Example 4

```
#    FIND NEXT RECORD EMPLOYEE OF SET DEPT-SET.
```

Assuming that the current record of run-unit is Employee 1, the record chosen by this format of the record selection expression would be Employee 3. This record would then become current of run-unit, etc.

Example 5

```
#    FIND RECORD DEPT.
```

Assuming the Department name field was assigned the value of the Dept name, the Department record (whose field matched the Dept name) would have been selected and would have become current of run-unit.

Example 6

```
#    FIND EMPLOYEE VIA DEPT-SET USING EMPNUM.
```

Assuming the set selection clause for Dept Set is location mode of owner and the call field for the Dept record was assigned to PROGRAMMING and the database-identifier EMPNUM was assigned the value 5, this format at the record selection expression would have first located the record Dept which had the DEPTNUM field with the value programming and then located the Employee record with the EMPNUM of 6, which was owned by that Department. The EMPLOYEE record would then become current of run-unit, current of record EMPLOYEE, current of set DEPT-SET, and current of area in which it is stored.

****
* REMOVE *
****

Function

To cancel the membership of the object record in the occurrences of the specified set-names in which it currently participates as a member, provided that the object record is defined as an optional member of the sets named.

General Format

$$\underline{\text{REMOVE}}\ \underline{\text{FROM}} \left\{ \begin{array}{l} \underline{\text{SETS}}\ \text{set-name-1}\ [\underline{,}\text{set-name-2}]\ldots \\ \underline{\text{ALL}}\ \text{SETS} \end{array} \right\}.$$

Syntax Rules

1. The Sets set-name-1, set-name-2 must be defined in the invoke subschema.

2. The record which is current of run-unit must be defined as an optional member of the named sets in the schema DDL.

General Rules

1. The object record of the REMOVE statement is the current record of the run-unit.

2. If set-names are specified in the REMOVE statement, then the object record must have been defined in the schema as an optional member of all of the sets named. It must also currently participate as a member in an occurrence of at least one of the sets named.

3. The ALL SETS format is designed to remove the object record from all set occurrences in which it is defined in the invoked subschema as an optional member and in which it currently participates as a member.

4. No change occurs to any currency information maintained by the DBMS.

5. If the run-unit has not satisfied the privacy locks for all records, sets, areas, and data items needed to execute the REMOVE statement, Error Status Condition 1104F will result and the run-unit will be aborted by DBMS.

6. In addition to the conditions described, if any of the following conditions are encountered, the REMOVE statement is not successfully executed, the database remains in the state existing prior to the attempted execution, and the appropriate Error Status Condition code is made available in special register ERSTAT. Otherwise, the contents of ERSTAT is set to zero, indicating successful execution of the REMOVE statement.

   - If all set-names specified are not included in the invoked subschema, Error Status Condition 1146 results.

   - If the current record of the run-unit is not known, Error Status Condition 1113 results.

   - If the object record is not defined as an OPTIONAL MEMBER of all of the specified set-names, Error Status Condition 1115 results.

   - If the object record does not currently participate as a member in an occurrence of at least one of the sets specified or implied, Error Status Condition 1122 results.

   - If the object record or any record occurrence affected by the REMOVE statement is located in an area which is open for RETRIEVAL, Error Status Condition 1109 results.

   - If any record occurrence needed by the DBMS for informational purposes (such as following a search path) is not available because it is off-line or under exclusive control of another run-unit, Error Status Condition 1118 results.

Error Status Codes for REMOVE

Condition	Content of ERROR-STATUS
Incorrect usage-mode for area	1109
No current record of run-unit	1113
Object record not defined as an optional member of a named set	1115
Implicitly referenced area not available	1118
Object record not currently a member of named or implied set	1122
Concurrent update conflict	1135
Specified set-name not in subschema	1146
Fatal - privacy breach attempted	-1104(1104F)

Examples and Discussion for REMOVE

```
#     REMOVE FROM ALL SETS.
```

This format of the REMOVE statement will take the current record of run-unit and remove it from all sets of which it is an optional member. That is, the record will no longer be accessible from the set occurrences from which it has been removed. It will still be accessible from other set occurrences of which it is a mandatory member or from its CALC key if the record has a location mode of CALL or from a traversal of the AREA.

****
* START OF TRANSACTION *
****

Function

The START TRANSACTION command defines the start of either an update or retrieval activity on the database. It provides a retrieval transaction a consistent view of the database. It allows an update transaction to define the beginning of a recoverable series of commands and it guarantees exclusive update rights to data which may be modified. It also saves the currency indicators as they are defined at the beginning of the transaction.

General Format

START TRANSACTION identifier [**UPDATE** / **RETRIEVAL**].

Syntax Rules

1. Identifier is a level 77 COBOL identifier which contains a COMPUTATIONAL value assigned to the transaction as the user's (run-unit's) name for that transaction.

General Rules

1. All commands which access the database (FIND, GET, FETCH, MODIFY, STORE, INSERT, REMOVE, DELETE, IF, MOVE) must be nested between START TRANSACTION, END OF TRANSACTION or ABORT TRANSACTION statements.

2. The following commands may not be nested between the START and END or ABORT TRANSACTION statement: INVOKE, OPEN, CLOSE, START TRANSACTION and EXIT DBMS.

3. If Before-imaging is turned off by the DBACP, roll back cannot occur (i.e., a subsequent ABORT TRANSACTION will not roll back the DB).

4. If a subsequent update command (MODIFY, STORE, INSERT, REMOVE, DELETE) attempts to modify data that has been modified by another update transaction that is currently active, the update command will receive an Error Status of mm35 where mm is the major code. The only choice allowed the application at that point is to abort the transaction and start it over again. If this isn't done, on excution of the next DML command, the transaction will be rolled back and the run-unit aborted.

5. If a START TRANSACTION command is executed when a transaction is already active, ERROR STATUS 2435 results.

Example and Discussion of START OF TRANSACTION

```
       WORKING-STORAGE SECTION.
       .
       .
       .


       77  TRANID      COMP VALUE 1.
       77  ONE         COMP VALUE 1.
       .
       .

       PROCEDURE DIVISION.
       FIRST SECTION.
       BEGIN.
      #                INVOKE DBMS.
      #                OPEN ALL AREAS USAGE-MODE IS UPDATE.
       LABEL-1
           ADD ONE TO TRANID.
      #                START TRANSACTION TRANID, UPDATE.
      #                FIND RECORD DEPT.
      #                STORE EMPLOYEE.
           IF ERSTAT = Ø GO TO LABEL-1Ø.
      #                CLEAR ERROR.
      #                ABORT TRANSACTION TRANID.
           GO TO LABEL-1.
       LABEL-1Ø.
           .
           .
           .
      #                END TRANSACTION TRANID.
           .
           .
           .
           GO TO LABEL-1.
```

The above program is an example of a procedure that might be used to add an employee record to a department in a concurrent on-line environment.

The DBMS is first invoked and all areas defined in the subschema are opened. Then, a start of transaction is executed. During the transaction a FIND statement is executed and then a STORE. If the STORE statement has a concurrent update error, the transaction is rolled back and tried again. If it is successful, execution continues until the END TRANSACTION, which then secures all updates to disk and defines the start of a quiescent point until the next START TRANSACTION.

****
* STORE *
****

Function

The STORE statement accomplishes the following:

1. Acquires space and a database-key for a new record occurrence in the database.

2. Causes the values of the appropriate data items in User Work Area to be included in the occurrence of the object record in the database.

3. Inserts the object record into all sets for which it is defined as an automatic member in the schema.

4. Establishes a new set occurrence for each set where the object record is defined as owner in the schema.

5. Establishes the object record as the current record of the run-unit.

6. Depending on the SUPPRESS status, establishes the object as:

   - The current record of the area in which it is stored,
   - The current record of the record-name,
   - The current record of set for all set-names in which it is specified as an owner or automatic member.

General Format

STORE record-name.

Syntax Rules

1. The invoked subschema must include: The named record; the data items or sets specified in the LOCATION MODE clause of the named record; at least one of the areas specified in the WITHIN clause of the named record, all sets in which the named record is defined as an owner or an automatic member; all data items, records, and sets specified or referenced in the SET OCCURRENCE SELECTION clauses and ASCENDING/DESCENDING KEY clauses of those sets in which the named record is defined as an automatic member.

General Rules

1. The subschema invoked must include the named record; otherwise, Error Status Condition 1246 results.

2. The data items, data-base-data-names, or sets specified in the LOCATION MODE clause of the named record must be included in the invoked subschema; otherwise, Error Status Condition 1243 results.

3. The area in which the record is to be stored must be included in the invoked subschema; otherwise, Error Status Condition 1247 results.

4. All sets in which the named record is defined as an automatic member must be included in the invoked subschema; otherwise, Error Status Condition 1208 results.

5. All data items, data-base-data-name, records, and sets specified or referenced in the SET OCCURRENCE SELECTION clauses of those sets in which the named record is defined as an automatic member must be included in the subschema; otherwise, Error Status Condition 1244 results.

6. All data items specified in the ASCENDING/DESCENDING KEY clauses of those sets in which the named record is defined as an automatic member must be included in the invoked subschema; otherwise, Error Status Condition 1248 results.

7. A database-key and space for the object record are allocated on the basis of the description of the record in the subschema invoked by the run-unit and values provided by the user.

   - Data items from the User Work Area are included in the object record of the STORE statement. Data items not initialized will be given indeterminate values.

   - Also included is any control data item, as specified or referenced in the SET OCCURRENCE SELECTION clauses for this record, when all sets are defined as automatic members. If any such SET OCCURRENCE SELECTION clause is THRU CURRENT SET, the user must insure that the current record of that set identifies the proper set occurrence. If any such SET OCCURRENCE SELECTION clause is THRU LOCATION MODE OF OWNER, then all control data items specified for that owner record must be initialized. In addition, all other data items specified in the applicable SET OCCURRENCE SELECTION clauses must be initialized.

8. For each LOCATION MODE clause involved in the execution of a STORE, the following control data items (dbdn's) (depending on the option used) must be initialized.

- In the case of the DIRECT phrase, the dbdn specified in the LOCATION MODE clause must be initialized. In a STORE statement, the first 28-bits of such a data item is ignored. The last two characters of the data-field must be initialized with zero or a positive integer number. If a non-zero value is given, this word will be used by the DBMS as the Bucket-number portion of the database-key. If a zero has been used, DBMS will assign an available Bucket number for the record type in the specified area. In either case, the DBMS will generate the first 20-bits of the database-key (which represents the record type and the area in which the occurrence is to be stored). The system will assign the resulting 48-bit DBK, if it is available, to the record which is the object of the STORE statement. If the occurrence-number portion was supplied by the user, and the resulting database-key is not available, the DBMS will assign an occurrence-number which will result in an available DBK in the specified area. In any case, the full 48-bit database-key by which the record occurrence is stored will be returned in the dbdn specified in the DIRECT phrase of the LOCATION MODE clause.

  A database-key can be determined by the DBMS to be 'not available' for either of the following reasons:

  - The database-key is already in use either by a current or a deleted record occurrence of the database.

  - The storage structure implemented under the current version of the DBMS will not allow the database-key, since it will cause overflow.

    If the DIRECT phrase is used, and the WITHIN clause is specified, the dbdn declared as AREA-ID must be initialized with an area-name specified or implied for the record type or to zero, in which case the first area in which the record definition is included in the schema will be chosen. If an area-name is specified which is not defined in the schema as a possible area for this record type, Error Status Condition 1223 results.

    - In the case of the CALC phrase, the database-identifiers named in the LOCATION MODE clause must be initialized, and if the record type can be stored in more than one area, the dbdn declared as AREA-ID in the WITHIN clause must be initialized with an area-name specified or implied for the record type (or zero, as above). If an area-name is specified which is not defined in the schema as a possible area for the record type, Error Status Condition 1223 results. The DBMS will develop a database-key which is compatible with the specified area-name.

- In the case of the VIA set-name phrase, the data items specified in the SET OCCURRENCE SELECTION clause for this record in the named set must be initialized, and, if the record type can be stored in more than one area, the dbdn declared as AREA-ID in the WITHIN clause must be initialized with an area-name specified or implied for the record type (or zero, as above). Initialization of the data items specified in the SET OCCURRENCE SELECTION clause is required to enable the DBMS to select a unique occurrence of the named set, and is required regardless of whether the record being stored is an automatic or manual member of that set. If it is an automatic member of the set, it will be logically inserted into the selected set occurrence. If it is a manual member, it will not be inserted into the selected set occurrence. In both cases, however, subject to the constraints of the implied area and areas specified in the WITHIN clause, the record being stored will be placed by the DBMS as close as is possible to the actual or probable logical insert point in the selected set occurrence. The effect of implicit and explicit areas on placement is as follows: if the record type is defined in the schema to be a possible occurrence of only one area-name, placement is controlled by that area-name; If more than one area-name is specified, the initialized value of AREA-ID controls placement. AREA-ID must be initialized with an area-name appropriate for the record type or it must be initialized with a zero; otherwise, Error Status Condition 1223 results. If AREA-ID is initialized with a legal area-name, placement is in that area. If AREA-ID is initialized with a zero, placement occurs as close as is possible to the logical insert point. Such placement is, however, constrained by the implied or specified areas for this record.

9. If no LOCATION MODE clause has been specified for the record being stored, DBMS will assign a database-key consistent with the relevant area-name. If the record can be stored in more than one area, AREA-ID specified in the WITHIN clause must be initialized with an area-name specified or implied for the record type or zero (as above). If an area-name is specified which is not included for the record type, Error Status Condition 1223 results.

10. In cases where the type of a data item, as defined in the subschema invoked by the run-unit, differs from the definition of that data item in the schema:

    - Conversions performed by the DBMS will be in accordance with the rules specified in the TYPE clause of the DDL for the SCHEMA and the PICTURE and USAGE specifications of the subschema DDL for COBOL.

11. Data items defined in the schema for the database but not included in the subschema invoked by the run-unit are not assigned User Work Area locations. Null-values will be placed in the database for such data items.

12. The object record occurrence is inserted into a set occurrence for each set in which the record is defined as an automatic member. The ordering rules for the set govern the insertion point of the object record in all of the relevant set occurrences.

13. The object record is established as the owner of a set occurrence for each set in which it has been defined as an owner. These set occurrences are empty at this time; that is, they have no member records.

14. The successfully stored record occurrence becomes the current record of the run-unit.

15. If suppression of currency updates is not in effect (see SUPPRESS command), the object record also becomes the current record of the area in which it is placed, the current record of its record-name, and the current record of all set-names in which it is defined as an owner or automatic member.

16. The SUPPRESS command provides the selective facility to prevent the object record from becoming the current record of the area, the current record of its record-name, and the current record of any or all of the set-names in which it is defined as either an owner or an automatic member. When SUPPRESS is used, the specified currency indicators are not affected by the execution of the STORE statement.

    Use of SUPPRESS cannot prevent the object record from becoming the current record of the run-unit.

17. If any of the following conditions are encountered, the STORE statement is not successfully executed, the database remains in the state existing prior to the attempted execution, and the appropriate Error Status Condition code is made available through the special register ERSTAT. Otherwise, ERSTAT will be zero, indicating successful execution of the STORE statement.

- If the named record is not included in the invoked subschema, Error Status Condition 1246 results.

- If all database entities specified in the LOCATION MODE clause of the named record are not included in the invoked subschema, Error Status Condition 1243 results.

- If the area in which the named record is to be stored is not included in the invoked subschema, Error Status Condition 1247 results.

- If the object record is to be stored in an area which is not open, Error Status Condition 1201 results.

- If the object record or any record occurrence affected by the STORE statement is located in an area which is only open for retrieval, Error Status Condition 1209 results.

- If any record occurrence needed by the DBMS for informational purposes (such as following a search path) is not available because it is off-line or under the exclusive control of another run-unit, Error Status Condition 1218 results.

- If media space is not available, (i.e., there is insufficient room in the area specified or in a set where INSERTion would occur for the new record occurrence), Error Status Condition 1211 results.

- If the area specified for the object record, or for any record selected through its CALC keys as a result of the execution of the STORE statement, is not one of those implicitly or explicitly specified in the schema definition of the relevant record type, Error Status Condition 1223 results.

- If the record to be stored would violate a DUPLICATES NOT ALLOWED clause defined for any of the records or sets involved, Error Status Condition 1205 results.

- If all sets in which the named record is defined as an automatic member are not included in the invoked subschema, Error Status Condition 1208 results.

- If for any of the set-names involved, a set occurrence which meets the relevant set selection criterion cannot be found, Error Status Condition 1225 results.

- Error Status Condition 1244 results if all database entities specified or referenced in the SET OCCURRENCE SELECTION clauses of those sets in which the named record is defined as an automatic member are not included in the invoked subschema.

- If all data items specified in the ASCENDING/DESCENDING KEY clauses or search key clauses of those sets in which the named record is defined as an automatic member are not included in the invoked subschema, Error Status Condition 1248 results.

- If a CHECK clause applies and any of the stored data items is detected as invalid, Error Status Condition 1227 results.

- If the value of the data item in the User Work Area is such that it cannot be converted to the format specified in the schema for that data item, Error Status Condition 1219 occurs.

18. If the run-unit has not satisfied the privacy locks on all data items, records, sets, and areas needed to execute the STORE, Error Status Condition 12Ø4F will result and the run-unit will be aborted by the DBMS.

Error Status Codes for the STORE Statement

Condition	Content of ERROR STATUS
Area not open	1201
Violation of DUPLICATES NOT ALLOWED clause	1205
Referenced set-name not in subschema	1208
Incorrect usage-mode for area	1209
Media space not available	1211
Implicitly referenced area not available	1218
Conversion of value of data item not possible	1219
Illegal area-name	1223
No set occurrence satisfies argument values specified	1225
New value of data item or actual result data item violates CHECK clause	1227
Concurrent Update Error	1235
Arguments of Location Mode clause not included in subschema	1243
Arguments of Set Occurrence Selection clause not included in subschema	1244
Specified data item, record, set, or area not in subschema	1246
Referenced area-name not in subschema	1247
Arguments of ASCENDING/DESCENDING KEY clause not in subschema	1248
Fatal - privacy breach attempted	-1204(1204F)

Examples and Discussion for the STORE Statement

```
#    STORE DEPT.
```

The STORE command creates the record in the database, assigns it a DBK, inserts it into all sets for which it is an automatic member, and creates entries for all sets which it owns. Data items which are declared in the schema but not defined in the subschema are filled with null values.

****
* SUBSCHEMA *
****

Function

To name the subschema which will provide the description of the data to the application program.

General Format

SUBSCHEMA subschema-name of SCHEMA schema-name.

Syntax Rules

1. The schema-name specified must be the name of a schema known to the DBMS.

2. The subschema-name specified must refer to a COBOL subschema for the named schema and must be known to the system.

3. This statement must be present in any program unit (main program or subprogram) containing CDML commands. It must appear somewhere in the Working Storage section in the Data Division.

4. If there is more than one program unit in a file, the SUBSCHEMA statements in those program units must all refer to the same schema-name and subschema-name. This error cannot be defected by the DBMS, and must therefore be verified by the user.

General Rules

1. When the COBOL program is preprocessed, CDML will insert declarations for all records, data-base-identifiers and data-base-data-names defined in the specified subschema. The COBOL programmer must therefore ensure that other identifiers declared in the program do not have naming conflicts with the subschema names.

Example and Discussion of the SUBSCHEMA Statement

```
 WORKING-STORAGE SECTION.
#                    SUBSCHEMA MY-SUB OF SCHEMA THE-SCHEMA.
 Ø1  DUMMY-REC       PIC X(1Ø).
 77  DUMMY-VAL       COMP VALUE 1.
     .
     .
     .
```

The above example shows how a sample application program might appear in a file. This program consists of a main routine and two subroutines which will to reference the database.

The above statement is replaced by the Working Storage declarations which serve as the User Work Area.

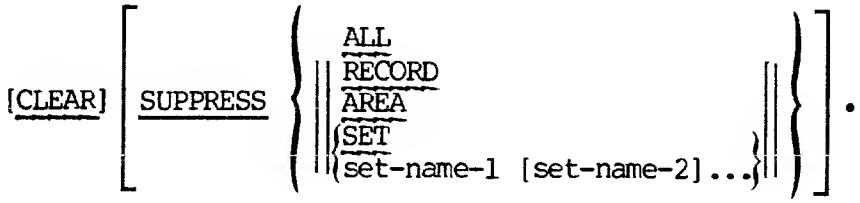
****
* SUPPRESS *
****

Function

Suppresses, that is prevents, the updating of any combination of the currency indicators: Current record of areas, current record of records, and current record of sets, or optionally, current record of specific sets.

Optionally, it clears all suppression which exists and indicates new suppression of currency indicators.

General Format



Syntax Rules

1. The Sets set-name-1, set-name-2 must be defined in the invoked subschema.

General Rules

1. The SUPPRESS statement will dictate the currency indicator updates performed by succeeding FIND, STORE, FETCH, or INSERT commands.

2. If the CLEAR option is not used, all currency indicators specified in the SUPPRESS statement are in addition to any currency indicators specified for suppression previously.

3. If the CLEAR option is used, the suppression now existing on any currency indicators will be lifted. Then, if the SUPPRESS phrase is also used, the specified currency indicators will be set for suppression of currency updates.

4. If all set-names specified are not included in the invoked subschema, Error Status Condition 1546 results; otherwise, special register ERSTAT is set zero, indicating successful execution of the SUPPRESS statement.

Error Status Condition of the SUPPRESS Statement

Condition	Content of ERSTAT
Set-name specified not in subschema	1546

Example and Discussion of the SUPPRESS Statement

```
          .
          .
          .
          .
#   FIND FIRST RECORD EMPLOYEE OF SET DEPT-SET.
          .
          .
#   SUPPRESS SET CURRENCY UPDATES.
          .
          .
#   FIND NEXT RECORD EMPLOYEE OF SET DEPT-SET.

#   CLEAR.

#   FIND NEXT RECORD EMPLOYEE OF SET DEPT-SET.

#   FIND NEXT RECORD EMPLOYEE OF SET DEPT-SET.
```

This program is an example of the use of the SUPPRESS statement. The first FIND makes the object record current of AREA, RECORD, and all sets which it owns or of which it is currently a member. The subsequent SUPPRESS command will turn off all currency updates for any set. The next FIND will update current of run-unit, current of area, and current of record, but will not update any set currency pointers. Thus, any reference to the current of set by an INSERT or FIND command will reference the set currency indicators defined by the first FIND.

In the example, the object record of the third FIND would be the same as the object record of the second FIND because the current of set DEPT-SET was not updated. After the CLEAR command is executed, the object record of the fourth FIND would be the third record of the set because the third FIND updated the currency pointer of that set.

SECTION 4

COBOL DDL & DML DIAGNOSTIC METHODS

SUMMARY OF MAJOR CODES

Major Code	DML Statement
Ø1	CLOSE
Ø2	DELETE
Ø3	FIND
Ø5	GET
Ø7	INSERT
Ø8	MODIFY
Ø9	OPEN
1Ø	CLEAR ERROR EXIT DBMS
11	REMOVE
12	STORE
13	PRIVACY KEY
14	INVOKE
15	SUPPRESS
18	IF
19	MOVE
22	FETCH
24	START TRANSACTION
25	END TRANSACTION
26	ABORT TRANSACTION

NOTE

ERROR CODE FORMAT
is XXYY where
major code is XX
and error code is YY

SUMMARY OF NON-FATAL MINOR CODES

Minor Code	Error Condition
01	Area not open
02	Database-key inconsistent with area-name
03	Invoke has already been executed
04	Data items invalid or inconsistent
05	Violation of DUPLICATES NOT ALLOWED clause
06	Current of set, area, or record-name not known
07	End of set or area
08	Referenced record or set-name not in subschema
09	Incorrect usage-mode for area
10	Not used
11	Media space not available
12	Database-key not available
13	No current record of run-unit
14	Object record is mandatory automatic in named set
15	Object record is mandatory in named set
16	Record already a member of named set
17	Deleted record involved
18	Implicitly referenced area not available
19	Conversion of value of data item not possible
20	Not used
21	Not used, no current owner record of named set
22	Record not currently a member of named set
23	Illegal area-name
24	Not used

25 No set occurrence satisfies argument values

26 No record satisfies the FIND specified

27 CHECK clause violation

28 Not used

29 Violation of deadlock protection rule

30 Unqualified delete attempted on non-empty set

31 OPEN blocked

32 Deleted set occurrence involved

33 Attempted to find owner of a singular set

34 Time out

35 Concurrent update conflict, refer to 35A

36 Not used

37 Not used

38 Not used

39 Not used

40 Record type not a member of named set

41 Record type not owner of named set

42 Record type not included in named area

43 Arguments of LOCATION MODE clause not included in subschema

44 Arguments of SET OCCURRENCE SELECTION clause not included in subschema

45 Location mode of record not specified as CALC

46 Specified data item, record, set or area not in subschema

47 Referenced area-name not included in subschema

48 Arguments of ASCENDING/DESCENDING clause not in subschema

49 Arguments of within clause not included in subschema

50 Not used

51	Not used
52	Virtual data item not available, not used
53	Invalid value for virtual result data item, not used
54	Value of string data item truncated in User Work Area
Ø1F	Required subschema has been deleted
Ø2F	Required schema has been deleted
Ø3F	Not used
Ø4F	Privacy breach attempted
Ø5F	Attempted to open too many areas simultaneously
Ø6F	Not used
Ø7F	Not used
Ø8F	Not used
Ø9F	Feature not implemented
1ØF	Internal fatal error
11F	Schema locked
12F	Database files not allocated
13F	Volume not on system
14F	Too many files opened on schemas in concurrent use
15F	User Work Area larger than declared space in dynamic invoke

NOTE
Fatal errors are indicated by a negative number. (e.g., - Ø311 is a fatal error for the FIND command).

SUMMARY OF CONCURRENT ACCESS CONFLICTS

Message	Meaning
1) ATTEMPTED ACCESS TO CONCURRENTLY UPDATED BLOCK	External conditions
2) ATTEMPTED ACCESS TO INVALID BEFORE-IMAGE	Slow reader - requested before image has been recycled
3) TRANSACTION MUST ABORT	A concurrency error has already occurred.
4) NO TRANSACTION IS IN EFFECT	DML program error.
5) OVERFLOW OF BIT MAP	Too many updates or a slow update or an aborted update.
6) OVERFLOW OF OLDEST BEFORE-IMAGE TABLE	Same as above.
7) OVERFLOW OF GENERATION NUMBER	More than 232 transactions (update transactions).
8) OVERFLOW OF BEFORE-IMAGE FILE	Too much update activity.
9) ATTEMPTED WRITE BY READ TRANSACTION	DML program error.
10) INVALID START UPDATE WITHIN ACTIVE TRANSACTION	DML program error.
11) INVALID START RETRIEVAL WITHIN ACTIVE TRANSACTION	DML program error.
12) INVALID ABORT TRANSACTION -- NO TRANSACTION ACTIVE	DML program error.
13) INVALID END TRANSACTION -- NO TRANSACTION ACTIVE	DML program error.
14) OPEN IS INVALID WITHIN ACTIVE TRANSACTION	DML program error.
15) CLOSE OR EXIT DBMS IS INVALID WITHIN ACTIVE TRANSACTION	DML program error.
16) CANNOT READ OR WRITE BEFORE, LOG OR AFTER FILE	DBMS error.
17) LOG FILE NOT OPEN	Can't happen currently.

18) OVERFLOW OF READ TRANSACTION NUMBER	More than 232 read transactions.
19) BEFORE-IMAGE FILE NOT OPEN	DBA error.
20) INCONSISTENT TRANSACTION IDENTIFIER	Identifier on END or ABORT TRANSACTION does not match identifier on START OF TRANSACTION.

The following error conditions are transient: 1,2,3,5,6,8; transient errors will eventually go away if the transaction is repeated. The following error conditions are fatal to a subsequent retry of the same transaction: 4,7,9,10,11,14,15,18,19.

INDEX